How to solve Basic Engineering and Mathematics Problems using Mathematica, Matlab and Maple

Nasser M. Abbasi

January 16, 2024 Compiled on January 16, 2024 at 3:50am [public]

Contents

This is a collection of how to examples showing the use of Mathematica and Matlab to solve basic engineering and mathematics problems. Few examples are also in Maple, Ada, and Fortran.

This was started as a cheat sheet few years ago, and I continue to update it all the time.

Few of the Matlab examples require the use of toolboxs such as signal processing toolbox and the control systems toolbox (these come free as part of the student version). Most examples require only the basic system installation.

CHAPTER 1

Control systems, Linear systems, transfer functions, state space related problems

1.1 Creating tf and state space and different Conversion of forms

1.1.1 Create continuous time transfer function given the poles, zeros and gain

Problem: Find the transfer function $H(s)$ given zeros $s = -1, s = -2$, poles $s = 0, s = -4, s = -6$ and gain 5.

1.1.1.1 Mathematica

```
Clear["Global`*"];
num  = (s+1)(s+2);
den  = (s)(s+4)(s+6);
gain = 5;
sys  = TransferFunctionModel[
             gain (num/den),s]
```

```
Out[30]= TransferFunctionModel[{
     {{5*(1 + s)*(2 + s)}},
     s*(4 + s)*(6 + s)}, s]
```

1.1.1.2 Matlab

```
clear all;
m_zeros = [-1 -2];
m_poles = [0 -4 -6];
gain  = 5;
sys   = zpk(m_zeros,m_poles,gain);
[num,den] = tfdata(sys,'v');
printsys(num,den,'s')
```

```
num/den =
    5 s^2 + 15 s + 10
   -------------------
   s^3 + 10 s^2 + 24 s
```

1.1.1.3 Maple

```
restart;
alias(DS=DynamicSystems):
zeros :=[-1,-2]:
poles :=[0,-4,-6]:
gain  :=5:
sys   :=DS:-TransferFunction(zeros,poles,gain):
#print overview of the system object
DS:-PrintSystem(sys);
```

```
exports(sys);
#To see fields in the sys object, do the following
# tf, inputcount, outputcount, statecount, sampletime,
# discrete, systemname, inputvariable, outputvariable,
# statevariable, parameters, systemtype, ModulePrint
tf:=sys[tf]; #reads the transfer function
Matrix(1,1,{(1,1)=(5*s^2+15*s+10)/(s^3+10*s^2+24*s)})
numer(tf[1,1]); #get the numerator of the tf
```

$$\mathrm{tf} = \left[\ \frac{5\,s^2+15\,s+10}{s^3+10\,s^2+24\,s}\ \right]$$

```
denom(tf[1,1]); #get the denominator of the tf
```

$$s * (s^2 + 10 * s + 24)$$

1.1.2 Convert transfer function to state space representation

1.1.2.1 problem 1

Problem: Find the state space representation for the continuous time system defined by the transfer function

$$H(s) = \frac{5s}{s^2 + 4s + 25}$$

1.1.2.2 Mathematica

```
Clear["Global`*"];
sys = TransferFunctionModel[(5 s)/(s^2+4 s+25),s];
ss  = MinimalStateSpaceModel[StateSpaceModel[sys]]
```

Out[48]= $\left(\begin{array}{cc|c} 0 & 1 & 0 \\ -25 & -4 & 1 \\ \hline 0 & 5 & 0 \end{array}\right)$ $\mathcal{S}$

```
(a=Normal[ss][[1]])//MatrixForm
(b=Normal[ss][[2]])//MatrixForm
(c=Normal[ss][[3]])//MatrixForm
(d=Normal[ss][[4]])//MatrixForm
```

Out[49]//MatrixForm=

$$\begin{pmatrix} 0 & 1 \\ -25 & -4 \end{pmatrix}$$

Out[50]//MatrixForm=

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Out[51]//MatrixForm=

$$\begin{pmatrix} 0 & 5 \end{pmatrix}$$

Out[52]//MatrixForm=

$$\begin{pmatrix} 0 \end{pmatrix}$$

1.1.2.3 Matlab

```
clear all;
s      = tf('s');
sys_tf = (5*s) / (s^2 + 4*s + 25);
[num,den] = tfdata(sys_tf,'v');
[A,B,C,D] = tf2ss(num,den)
```

```
A =
    -4   -25
     1     0
B =
     1
     0
C =
     5     0
D =
     0
```

1.1.2.4 Maple

```
restart;
alias(DS=DynamicSystems):
sys := DS:-TransferFunction(5*s/(s^2+4*s+25)):
sys := DS:-StateSpace(sys);
exports(sys); #to see what fields there are
a := sys:-a;
```

$$\begin{bmatrix} 0 & 1 \\ -25 & -4 \end{bmatrix}$$

```
b:=sys:-b;
```

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

```
c:=sys:-c;
```

$$\begin{bmatrix} 0 & 5 \end{bmatrix}$$

```
d:=sys:-d;
```

$$\begin{bmatrix} 0 \end{bmatrix}$$

1.1.2.5 problem 2

Problem: Given the continuous time S transfer function defined by

$$H(s) = \frac{1+s}{s^2+s+1}$$

convert to state space and back to transfer function.

Mathematica

```
Clear["Global`*"];
sys = TransferFunctionModel[(s + 1)/(s^2 + 1 s + 1), s];
ss = MinimalStateSpaceModel[StateSpaceModel[sys]]
```

$$\left(\begin{array}{cc|c} 0 & 1 & 0 \\ -1 & -1 & 1 \\ \hline 1 & 1 & 0 \end{array}\right)$$

```
TransferFunctionModel[ss]
```

$$\frac{s+1}{s^2+s+1}$$

Matlab

```
clear all;
s   = tf('s');
sys = ( s+1 )/(s^2  + s  +  1);
[num,den]=tfdata(sys,'v');
%convert to state space
[A,B,C,D] = tf2ss(num,den)
```

```
A =
    -1    -1
     1     0

B =
     1
     0

C =
     1     1

D =
     0
```

```
%convert from ss to tf
[num,den] = ss2tf(A,B,C,D);
sys=tf(num,den)
```

```
sys =
     s + 1
  -----------
  s^2 + s + 1
```

1.1.3 Create a state space representation from A,B,C,D and find the step response

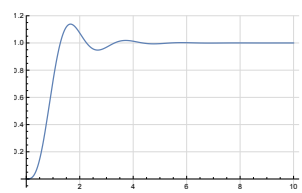
Problem: Find the state space representation and the step response of the continuous time system defined by the Matrices A,B,C,D as shown

```
A =
     0     1     0     0
     0     0     1     0
     0     0     0     1
  -100   -80   -32    -8

B =
     0
     0
     5
```

```
    60

C =
     1     0     0     0

D =
     0
```

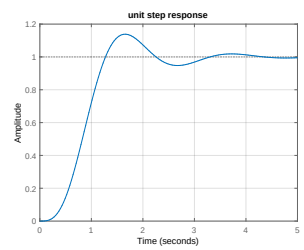
Mathematica

```
Clear["Global`*"];

a = {{0,1,0,0},
     {0,0,1,0},
     {0,0,0,1},
     {-100,-80,-32,-8}
    };

b = {{0} ,{0}, {5},{60}};
c = {{1,0,0,0}};
d = {{0}};

sys = StateSpaceModel[{a,b,c,d}];
y   = OutputResponse[sys,UnitStep[t],{t,0,10}];

p = Plot[Evaluate@y, {t, 0, 10},
         PlotRange -> All,
         GridLines -> Automatic,
         GridLinesStyle -> LightGray]
```



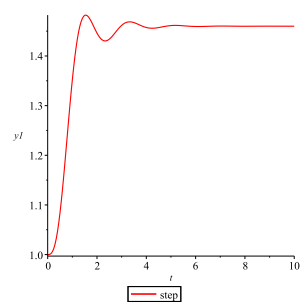
Matlab

```
clear all;
A = [0 1 0 0;
     0 0 1 0;
     0 0 0 1;
   -100 -80 -32 -8];
B = [0;0;5;60];
C = [1 0 0 0];
D = [0];

sys = ss(A,B,C,D);

step(sys);
grid;
title('unit step response');
```



Maple

```
restart:
alias(DS=DynamicSystems):
alias(size=LinearAlgebra:-Dimensions);
a:=Matrix([[0,1,0,0],[0,0,1,0],
           [0,0,0,1],[-100,-90,-32,-8]]):
b:=Matrix([[0],[0],[5],[6]]):
c:=Matrix([[1,0,0,0]]):
d:=Matrix([[1]]):
sys:=DS:-StateSpace(a,b,c,d);

DS:-ResponsePlot(sys, DS:-Step(),
  duration=10,color=red,legend="step");
```



1.1.4 Convert continuous time to discrete time transfer function, gain and phase margins

Problem: Compute the gain and phase margins of the open-loop discrete linear time system sampled from the continuous time S transfer function defined by

$$H(s) = \frac{1+s}{s^2+s+1}$$

Use sampling period of 0.1 seconds.

Mathematica

```
Clear["Global`*"];
tf = TransferFunctionModel[(s+1)/(s^2+1 s+1),s];
```

Out[88]= $\left(\frac{1+s}{1+s+s^2}\right)_{\mathcal{T}}$

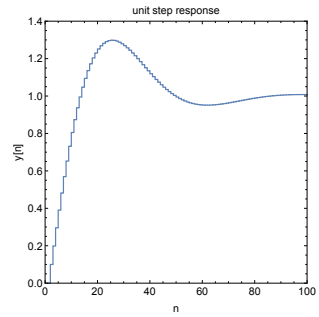
```
ds = ToDiscreteTimeModel[tf, 0.1, z,
                Method -> "ZeroOrderHold"]
```

Out[89]= $\left(\frac{-0.0903292+0.0998375\,z}{0.904837-1.89533\,z+z^2}\right)_{\mathcal{T}_{0.1}}$

```
{gm, pm} = GainPhaseMargins[ds];
```

```
gm
 Out[28]={{31.4159,19.9833}}
pm
 Out[29]={{1.41451,1.83932},{0.,Pi}}
```

```
max = 100;
yn  = OutputResponse[ds,
          Table[UnitStep[n],{n, 0, max}]];
ListPlot[yn,
      Joined -> True,
      InterpolationOrder -> 0,
      PlotRange -> {{0, max}, {0, 1.4}},
      Frame -> True,
      FrameLabel ->{{"y[n]", None},
                    {"n", "unit step response"}},
      ImageSize -> 400,
      AspectRatio -> 1,BaseStyle -> 14]
```

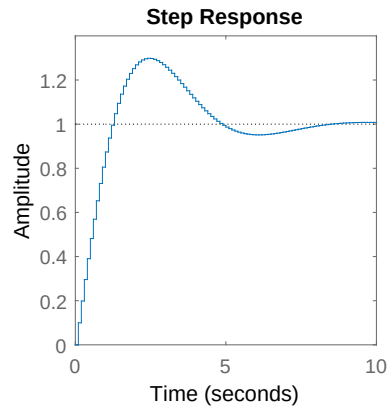


Matlab

```
clear all; close all;
Ts   = 0.1; %sec, sample period
s    = tf('s');
sys  = ( s + 1 ) / (  s^2  + s  +  1);
%convert s to z domain
sysd = c2d(sys,Ts,'zoh');
tf(sysd)
```

```
  0.09984 z - 0.09033
 ----------------------
 z^2 - 1.895 z + 0.9048

Sample time: 0.1 seconds
Discrete-time transfer function.

Gm =
   19.9833

Pm =
  105.3851

Wcg =
   31.4159

Wcp =
    1.4145
```

```
step(sysd,10);
[Gm,Pm,Wcg,Wcp] = margin(sysd)
```



1.1.5 Convert differential equation to transfer functions and to state space

Problem: Obtain the transfer and state space representation for the differential equation

$$3\frac{d^2y}{dt^2} + 2\frac{dy}{dt} + y(t) = u(t)$$

Mathematica

```
Clear[y, u, t];
eq =  3  y''[t] + 2 y'[t] +  y[t] == u[t];
ss = StateSpaceModel[ {eq}, {{y'[t], 0},
        {y[t], 0}}, {{u[t], 0}}, {y[t]}, t];
ss = MinimalStateSpaceModel[ss]
```

Out[34]= $\left(\begin{array}{cc|c} -\frac{2}{3} & -\frac{1}{3} & \frac{1}{3} \\ 1 & 0 & 0 \\ \hline 0 & 1 & 0 \end{array}\right)_{\mathcal{S}}$

```
tf = Simplify[TransferFunctionModel[ss]]
```

Out[35]= $\left(\frac{1}{1 + 2\,s + 3\,s^2}\right)_{\mathcal{T}}$

Matlab

```
clear all;

syms y(t) Y
eq  = 3*diff(y,2)+2*diff(y)+y;
lap = laplace(eq);
lap = subs(lap,'y(0)',0);
lap = subs(lap,'D(y)(0)',0);
lap = subs(lap,'laplace(y(t),t,s)',Y);
H   = solve(lap-1,Y);
pretty(H)
```

```
        1
  --------------
     2
  3 s  + 2 s + 1
```

```
[num,den] = numden(H);
num = sym2poly(num);
den = sym2poly(den);
[A,B,C,D] = tf2ss(num,den)
```

```
A =
   -0.6667   -0.3333
    1.0000         0
B =
     1
     0
C =
         0    0.3333
D =
     0
```

Maple

```
restart;
alias(DS=DynamicSystems):

ode:=3*diff(y(t),t$2)+2*diff(y(t),t)+y(t)=
                              Heaviside(t):

sys:=DS:-DiffEquation(ode,
          'outputvariable'=[y(t)],
          'inputvariable'=[Heaviside(t)]):

sys:=DS:-TransferFunction(sys):
sys:-tf(1,1);
```

$$\frac{1}{3s^2+2\,s+1}$$

```
sys:=DS:-StateSpace(sys):
#show the state space matrices
{sys:-a,sys:-b,sys:-c,sys:-d};
```

$$\left\{\begin{bmatrix} 0 & 1 \\ -1/3 & -2/3 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1/3 & 0 \end{bmatrix}, \begin{bmatrix} 0 \end{bmatrix}\right\}$$

1.1.6 Convert from continuous transfer function to discrete time transfer function

Problem: Convert

$$H(s) = \frac{1}{s^2+10s+10}$$

To Z domain transfer function using sampling period of 0.01 seconds and using the zero order hold method.

Mathematica

```
Clear["Global`*"]
sys = TransferFunctionModel[
              1/(s^2 +10 s+20),s];

sysd= ToDiscreteTimeModel[sys,0.01,z,
        Method->"ZeroOrderHold"]
```

$$\left(\frac{0.0000467806 + 0.0000483662\,z}{0.904837 - 1.90293\,z + z^2}\right)_{\mathcal{T}_{0.01}}$$

Matlab

```
s    = tf('s');
sys  = 1/(s^2 + 10*s + 20);
T    = 0.01; %seconds
sysz = c2d(sys,T,'zoh')
```

```
sysz =

  4.837e-05 z + 4.678e-05
  -----------------------
  z^2 - 1.903 z + 0.9048

Sample time: 0.01 seconds
Discrete-time transfer function.
```

1.1.7 Convert a Laplace transfer function to an ordinary differential equation

Problem: Give a continuous time transfer function, show how to convert it to an ordinary differential equation. This method works for non-delay systems. The transfer function must be ratio of polynomials. For additional methods see this question at stackexchange

Mathematica

```
tfToDiff[tf_, s_, t_, y_, u_]:=
 Module[{rhs,lhs,n,m},
  rhs = Numerator[tf];
  lhs = Denominator[tf];
  rhs = rhs /. m_. s^n_. -> m D[u[t], {t, n}];
  lhs = lhs /. m_. s^n_. -> m D[y[t], {t, n}];
  lhs == rhs
  ]

tf = (5s)/(s^2+4s+25);
tf = C0 s/(R0 C0 s + 1);
eq=tfToDiff[tf, s, t, y, u]
```

```
25 + 4 y'[t]+y''[t]==5 u'[t]
```

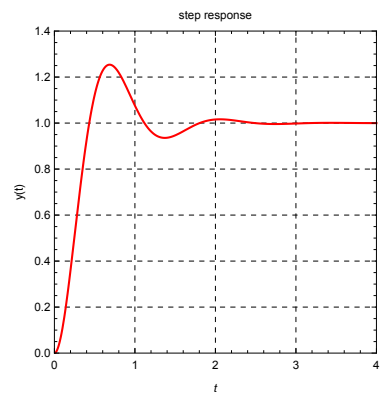
1.2 Obtain the step response of an LTI from its transfer function

Problem: Find the unit step response for the continuous time system defined by the transfer function

$$H(s) = \frac{25}{s^2 + 4s + 25}$$

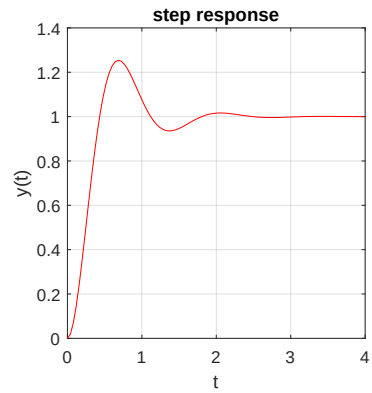
Mathematica

```
Clear["Global`*"];

tf=TransferFunctionModel[25/(s^2+4s+25),s];
y = OutputResponse[tf, UnitStep[t], t];

Plot[Evaluate@y, {t, 0, 4},
 PlotRange -> {{0, 4}, {0, 1.4}},
 Frame -> True,
 FrameLabel -> {{"y(t)", None},
                {t, "step response"}},
 GridLines -> Automatic,
 GridLinesStyle -> Dashed,
 ImageSize -> {300, 300},
 PlotStyle -> Red,
 AspectRatio -> 1]
```



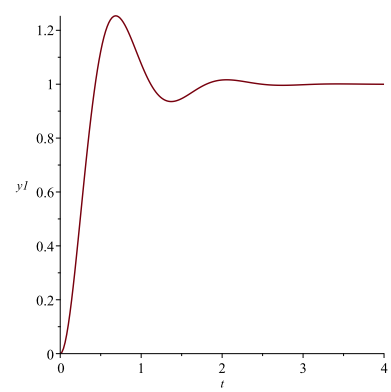
Matlab

```
clear all;
s     = tf('s');
sys   = 25/(s^2+4*s+25);
[y,t] = step(sys,4);

plot(t,y,'r');
xlim([0 4]);
ylim([0 1.4]);
title('step response');
xlabel('t');
ylabel('y(t)');
grid
set(gcf,'Position',[10,10,310,310]);
```



Maple

```
restart:
with(DynamicSystems):
sys := TransferFunction(25/(s^2+4*s+25)):
ResponsePlot(sys, Step(),duration=4);
```



1.3 plot the impulse and step responses of a system from its transfer function

Problem: Find the impulse and step responses for the continuous time system defined by the transfer function

$$H(s) = \frac{1}{s^2 + 0.2s + 1}$$

and display these on the same plot up to some t value.

Side note: It was easier to see the analytical form of the responses in Mathematica and Maple so it is given below the plot.

Mathematica

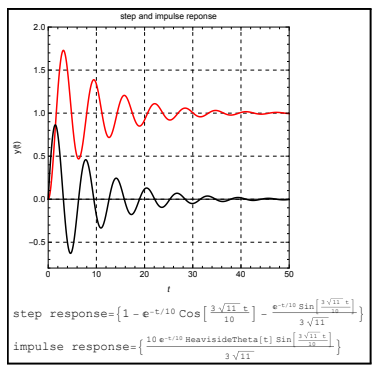
```
Clear["Global`*"];
SetDirectory[NotebookDirectory[]]
sys = TransferFunctionModel
        [1/(s^2 + 2/10 s + 1), s];

yStep = Assuming[t > 0,
  Simplify@OutputResponse[
             sys, UnitStep[t], t]]
```

```
{1 - E^(-t/10) Cos[(3 Sqrt[11] t)/10]
 - (
E^(-t/10) Sin[(3 Sqrt[11]t)/10])/
                          (3 Sqrt[11])}
```

```
yImpulse = Simplify@
  OutputResponse[sys,DiracDelta[t],t]
```

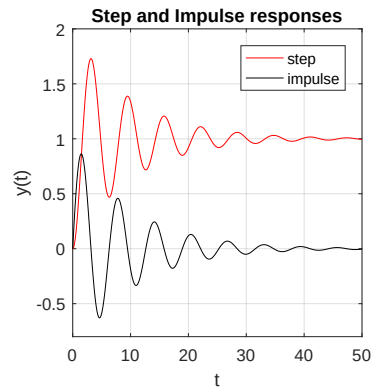
```
{(10 E^(-t/10) HeavisideTheta[t]
  Sin[(3 Sqrt[11] t)/10])/(3 Sqrt[11])}
```

```
p = Grid[{
  {Plot[Evaluate@{yStep, yImpulse},
   {t, 0, 50},
  PlotRange -> {{0, 50}, {-0.8, 2.0}},
  Frame -> True,
  FrameLabel -> {{"y(t)", None},
     {t, "step and impulse reponse"}},
  GridLines -> Automatic,
  GridLinesStyle -> Dashed,
  ImageSize -> {300, 300},
  PlotStyle -> {Red, Black},
  AspectRatio -> 1]
  },
  {Row[{"step response=", Re@yStep}]},
  {Row[{"impulse response=",Re@yImpulse}]}
  }, Alignment -> Left, Frame -> True]
```



Matlab

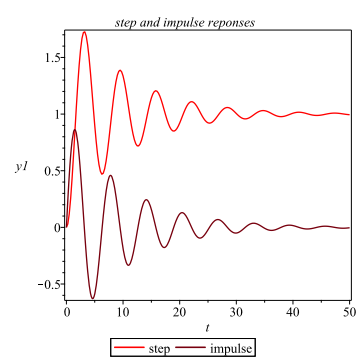
```
clear all; close all;
t   = 0:0.05:50;
s   = tf('s');
sys = 1/(s^2+0.2*s+1);
y   = step(sys,t);
plot(t,y,'-r')
hold on
y  = impulse(sys,t);
plot(t,y,'-k')
title('Step and Impulse responses');
xlabel('t');
ylabel('y(t)');
xlim([0 50]);
ylim([-0.8 2]);
legend('step','impulse');
grid on;
set(gcf,'Position',[10,10,310,310]);
```



Maple

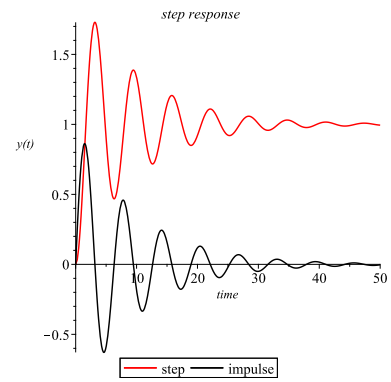
Using Maple DynamicSystems

```
restart:
alias(DS=DynamicSystems):
sys:=DS:-TransferFunction(1/(s^2+0.2*s+1)):
p1:=DS:-ResponsePlot(sys, DS:-Step(),
   duration=50,color=red,legend="step"):
p2:=DS:-ImpulseResponsePlot(sys,
                   50,
                   legend="impulse"):
plots:-display([p1,p2],axes=boxed,
   title=`step and impulse reponses`);
```



Using Laplace transform method:

```
with(inttrans):
with(plottools):
H:=1/(s^2+0.2*s+1);
input:=laplace(Heaviside(t),t,s):
yStep:=invlaplace(input*H,s,t);
input:=laplace(Dirac(t),t,s):
yImpulse:=invlaplace(input*H,s,t);
plot([yStep,yImpulse],t=0.001..50,
     title=`step response`,
     labels=[`time`,`y(t)`],
     color=[red,black],
     legend=["step","impulse"]);
```



1.4 Obtain the response of a transfer function for an arbitrary input

Problem: Find the response for the continuous time system defined by the transfer function

$$H(s) = \frac{1}{s^2 + 0.2s + 1}$$

when the input is given by

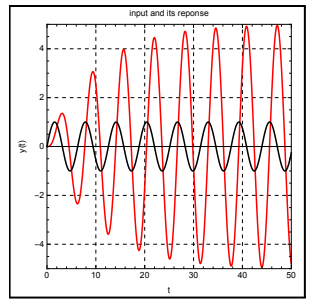
$$u(t) = \sin(t)$$

and display the response and input on the same plot.

Side note: It was easier to see the analytical form of the responses in Mathematica and Maple so it is given below the plot.

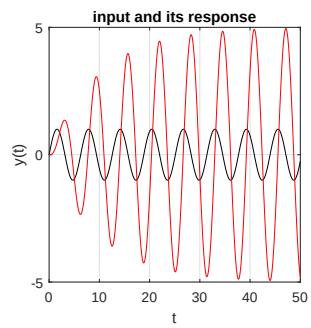
Mathematica

```
Clear["Global`*"];
SetDirectory[NotebookDirectory[]]
sys=TransferFunctionModel[1/(s^2+2/10 s+1),s];
u = Sin[t];
y = OutputResponse[sys, u, t];
p = Grid[{
   {Plot[Evaluate@{y, u}, {t, 0, 50},
     PlotRange -> {{0, 50}, {-5, 5}},
     Frame -> True,
     FrameLabel -> {{"y(t)", None},
           {"t", "input and its reponse"}},
     GridLines -> Automatic,
     GridLinesStyle -> Dashed,
     ImageSize -> {300, 300},
     PlotStyle -> {Red, Black},
     AspectRatio -> 1]
    }
   }, Alignment -> Left, Frame -> True]
```



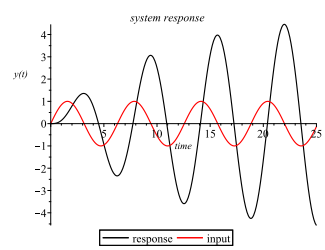
Matlab

```
clear all;
close all;
t   = 0:0.05:50;
u   = sin(t);
s   = tf('s');
sys = 1/(s^2+0.2*s+1);
[num,den] = tfdata(sys,'v');
y = lsim(num,den,u,t);
plot(t,u,'k',t,y,'r');
title('input and its response');
xlabel('t'); ylabel('y(t)');
xlim([0 50]);
ylim([-5 5]);
grid on;
set(gcf,'Position',[10,10,310,310]);
```



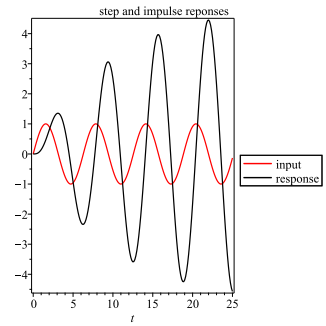
Maple

```
restart;
with(inttrans):H:=1/(s^2+0.2*s+1);
input:=sin(t);
inputS:=laplace(input,t,s):
y:=invlaplace(inputS*H,s,t);
plot([y,input],t=0..25,
 title=`system response`,
 labels=[`time`,`y(t)`],
 color=[black,red],
 legend=["response","input"]);
```



Using DynamicSystem package

```
restart:
alias(DS=DynamicSystems):
sys :=DS:-TransferFunction(1/(s^2+0.2*s+1)):
p1:=DS:-ResponsePlot(sys, sin(t),
 duration=25,color=black,legend="response"):
p2:=plot(sin(t),t=0..25,color=red,
   legend="input",size=[400,"default"]):
plots:-display([p2,p1],axes=boxed,
  title="step and impulse reponses",
  legendstyle= [location=right]);
```



1.5 Obtain the poles and zeros of a transfer function

Problem: Find the zeros, poles, and gain for the continuous time system defined by the transfer function

$$H(s) = \frac{25}{s^2 + 4s + 25}$$

Mathematica

```
Clear["Global`*"];
sys=TransferFunctionModel[25/(s^2+4 s+25),s];
TransferFunctionZeros[sys]
```

```
{{{}}}
```

```
TransferFunctionPoles[sys]//N
```

```
{{{-2.-4.58258 I,-2.+4.58258 I}}}
```

Matlab

```
clear all;
s = tf('s');
sys = 25/(s^2+4*s+25);
[z,p,k] =zpkdata(sys,'v')
```

```
z =
   Empty matrix: 0-by-1

p =
  -2.0000 + 4.5826i
  -2.0000 - 4.5826i
```

Maple

```
restart;
alias(DS=DynamicSystems):
sys:=DS:-TransferFunction(25/(s^2+4*s+25)):
r  :=DS:-ZeroPolePlot(sys,output=data):
zeros  := r[1];
poles  := r[2];
```

```
zeros:= []
poles:= [-2.000000000-4.582575695*I,
         -2.+4.582575695*I]
```

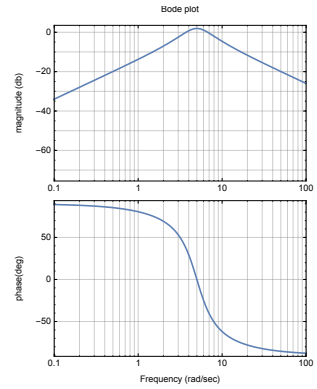
1.6 Generate Bode plot of a transfer function

Problem: Generate a Bode plot for the continuous time system defined by the transfer function

$$H(s) = \frac{5s}{s^2 + 4s + 25}$$

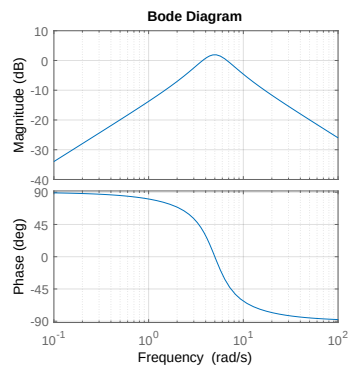
Mathematica

```
Clear["Global`*"];

tf=TransferFunctionModel[(5 s)/(s^2+4s+25),s];

BodePlot[tf, GridLines -> Automatic,
 ImageSize -> 300,
 FrameLabel -> {{{"magnitude (db)", None},
       {None, "Bode plot"}},
       {{"phase(deg)", None},
        {"Frequency (rad/sec)", None}}},
 ScalingFunctions -> {{"Log10", "dB"},
                      {"Log10", "Degree"}},
 PlotRange -> {{{0.1, 100}, Automatic},
               {{0.1, 100}, Automatic}}
 ]
```



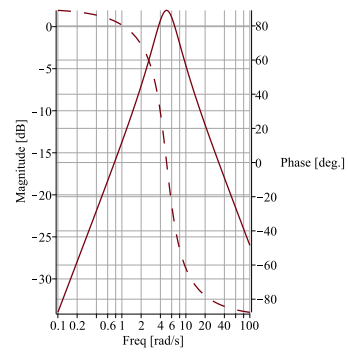
Matlab

```
clear all;
s   = tf('s');
sys = 5*s / (s^2 + 4*s + 25);
bode(sys);
grid;
set(gcf,'Position',[10,10,400,400]);
```



Maple

```
restart:
alias(DS=DynamicSystems):
sys:=DS:-TransferFunction(5*s/(s^2+4*s+25)):
DS:-BodePlot(sys,output=dualaxis,
             range=0.1..100);
```



or can plot the the two bode figures on top of each others as follows

```
DS:-BodePlot(sys,size=[400,300],
 output=verticalplot,range=0.1..100);
```

1.7 How to check that state space system $x' = Ax + Bu$ is controllable?

A system described by

$$x' = Ax + Bu$$
$$y = Cx + Du$$

Is controllable if for any initial state x_0 and any final state x_f there exist an input u which moves the system from x_0 to x_f in finite time. Only the matrix A and B are needed to decide on controllability. If the rank of

$$[B\ AB\ A^2B \dots\ A^{n-1}B]$$

is n which is the number of states, then the system is controllable. Given the matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 5 & 0 \end{pmatrix}$$

And

$$B = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -2 \end{pmatrix}$$

Mathematica

```
A0 = {{0, 1, 0, 0},
      {0, 0, -1, 0},
      {0, 0, 0, 1},
      {0, 0, 5, 0}};
B0 = {{0}, {1}, {0}, {-2}};
sys = StateSpaceModel[{A0, B0}];
m = ControllabilityMatrix[sys]
```

$$\begin{pmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \\ 0 & -2 & 0 & -10 \\ -2 & 0 & -10 & 0 \end{pmatrix}$$

```
ControllableModelQ[sys]
```

True

```
MatrixRank[m]
```

4

Matlab

```
A0 = [0 1 0 0;
      0 0 -1 0;
      0 0 0 1;
      0 0 5 0];
B0 = [0 1 0 -2]';
sys = ss(A0,B0,[],[]);
m   = ctrb(sys)
```

```
m =
     0     1     0     2
     1     0     2     0
     0    -2     0   -10
    -2     0   -10     0
```

```
rank(m)
```

4

Maple

```
restart:
alias(DS=DynamicSystems):
A:=Matrix( [ [0,1,0,0],
             [0,0,-1,0],
             [0,0,0,1],
             [0,0,5,0]
           ]
          );
B:=Matrix([[0],[1],[0],[-2]]);
sys:=DS:-StateSpace(A,B);
m:=DS:-ControllabilityMatrix(sys);
```

$$\begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \\ 0 & -2 & 0 & -10 \\ -2 & 0 & -10 & 0 \end{bmatrix}$$

```
DS:-Controllable(sys,method=rank);
```

true

```
DS:-Controllable(sys,method=staircase);
```

true

```
LinearAlgebra:-Rank(m);
```

4

1.8 Obtain partial-fraction expansion

Problem: Given the continuous time S transfer function defined by

$$H(s) = \frac{s^4 + 8s^3 + 16s^2 + 9s + 9}{s^3 + 6s^2 + 11s + 6}$$

obtain the partial-fractions decomposition.

Comment: Mathematica result is easier to see visually since the partial-fraction decomposition returned in a symbolic form.

Mathematica

```
Remove["Global`*"];
expr = (s^4+8 s^3+16 s^2+9 s+6)/
            (s^3+6 s^2+11 s+6);
Apart[expr]
```

```
2 + s +3/(1+s) -4/(2+s) -6/(3+s)
```

Matlab

```
clear all;
s=tf('s');
tf_sys = (s^4+8*s^3+16*s^2+9*s+6)/...
          (s^3+6*s^2+11*s+6);
[num,den] = tfdata(tf_sys,'v');
[r,p,k] = residue(num,den)
```

```
r =
   -6.0000
   -4.0000
    3.0000

p =
   -3.0000
   -2.0000
   -1.0000

k =
     1     2
```

Maple

```
p:=(s^4+8*s^3+16*s^2+9*s+9)/(s^3+6*s^2+11*s+6);
p0:=convert(p,parfrac);
```

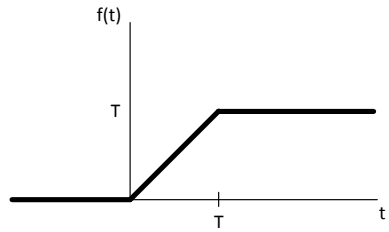
$$s+2-\frac{7}{s+2}-\frac{9}{2\,s+6}+\frac{9}{2\,s+2}$$

```
[op(p0)];
```

$$[s,2,\frac{7}{s+2},-\frac{9}{2\,s+6},\frac{9}{2\,s+2}]$$

1.9 Obtain Laplace transform for a piecewise functions

Problem: Obtain the Laplace transform for the function defined in the following figure.



Function f(t) to obtain its Laplace transform

Comment: Mathematica solution was easier than Matlab's. In Matlab the definition of the Laplace transform is applied to each piece separately and the result added. Not finding the piecewise maple function to access from inside MATLAB did not help.

Mathematica

```
Remove["Global`*"];
f[t_] := Piecewise[{{0,t<0},
                    {t,t>= 0 && t< T},
                    {T, t>T}}]
Simplify[LaplaceTransform[f[t],t,s] ,{T>0}]
```

```
Out[]= (1 - E^((-s)*T))/s^2
```

Matlab

```
clear all;
syms T t s;
syms s positive;
I1 = int(t*exp(-s*t),t,0,T);
I2 = int(T*exp(-s*t),t,T,Inf);
result = simple(I1+I2);
pretty(result)
```

```
  1 - exp(-T s)
  -------------
        2
       s
```

Maple

With Maple, had to use `Heaviside` else Laplace will not obtain the transform of a piecewise function.

```
restart;
assume(T>0):
interface(showassumed=0):
f:=t->piecewise(t<0,0,t>=0 and t<T,t,t>T,T):
r:=convert(f(t),Heaviside):
r:=inttrans[laplace](r,t,s);
```

$$\frac{1-e^{-sT}}{s^2}$$

1.10 Obtain Inverse Laplace transform of a transfer function

Problem: Obtain the inverse Laplace transform for the function

$$H(s)=\frac{s^4+5s^3+6s^2+9s+30}{s^4+6s^3+21s^2+46s+30}$$

Mathematica

```
Remove["Global`*"];
f = (s^4+5 s^3+6 s^2+9 s+30)/(s^4+6 s^3+21 s^2+46 s+30);
InverseLaplaceTransform[f,s,t];
Expand[FullSimplify[%]]
```

$$\delta(t)+\left(\frac{1}{234}+\frac{i}{234}\right)e^{(-1-3i)t}\left((73+326i)e^{6it}+(-326-73i)\right)-\frac{3e^{-3t}}{26}+\frac{23e^{-t}}{18}$$

Matlab

```
clear all;
syms s t
f = (s^4+5*s^3+6*s^2+9*s+30)/(s^4+6*s^3+21*s^2+46*s+30);
pretty(f)
```

```
    4       3       2
```

```
 s  + 5 s  + 6 s  + 9 s + 30
----------------------------
 4      3       2
s  + 6 s  + 21 s  + 46 s + 30
```

```
pretty(ilaplace(f))
```

```
                                                  /            399 sin(3 t) \
                                       253 exp(-t) | cos(3 t) + ------------ |
 23 exp(-t)   3 exp(-3 t)                          \                253      /
 ---------- - ----------- + dirac(t) - ----------------------------------------
     18           26                                        117
```

Maple

```
restart;
interface(showassumed=0):
p:=(s^4+5*s^3+6*s^2+9*s+30)/(s^4+6*s^3+21*s^2+46*s+30);
r:=inttrans[invlaplace](p,s,t);
```

$$Dirac(t) - \frac{3\,\mathrm{e}^{-3\,t}}{26} + \frac{(-506\,\cos(3\,t) - 798\,\sin(3\,t) + 299)\,\mathrm{e}^{-t}}{234}$$

1.11 Display the response to a unit step of an under, critically, and over damped system

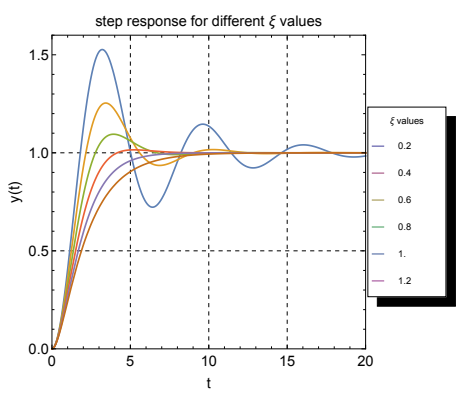
Problem: Obtain unit step response of the second order system given by the transfer function

$$H(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

in order to illustrate the response when the system is over, under, and critically damped. use $\omega_n = 1$ and change ξ over a range of values that extends from under damped to over damped.

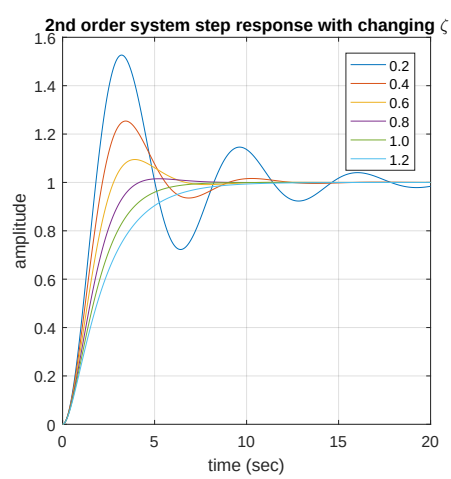
Mathematica

```
Clear["Global`*"];
Needs["PlotLegends`"]

sys=TransferFunctionModel[w^2/(s^2+2*z*w*s+w^2),s];
zValues = Range[.2,1.2,.2];
fun = OutputResponse[sys/.{w->1,z->#}, UnitStep[t],
       {t,0,20}]&/@zValues;

Plot[Evaluate@Flatten@Table[fun[[i]],
                          {i,1,Length[fun]}],
   {t,0,20},
   Frame->True,
   FrameLabel->{{"y(t)",None},
 {"t","step response for different \[Xi] values"}},
   PlotRange->{{0,20},{0,1.6}},
   GridLines->Automatic,
   GridLinesStyle->Dashed,
   PlotLegend->zValues,
   LegendPosition->{0.76,-0.5},LegendSize -> 1,
   LegendLabel -> "\[Xi] values",
   ImageSize -> 450,
   LabelStyle -> 14,AspectRatio -> 1
]
```



Matlab

```
clear; close all;
wn = 1;
z  = .2:.2:1.2;
t  = 0:0.05:20;
y  = zeros(length(t),length(z));
legendStr = cell(length(z),1);

for i = 1:length(z)
    [num,den] = ord2(wn,z(i));
    num = num*wn^2;
    [y(:,i),~,~] = step(num,den,t);
    legendStr(i) = {sprintf('%1.1f',z(i))};
end
plot(t,y);
legend(legendStr);

title(sprintf(
 '2nd order system step response with changing %s',
 '\zeta'));

xlabel('time (sec)');
ylabel('amplitude');
grid on;
set(gcf,'Position',[10,10,400,400]);
```



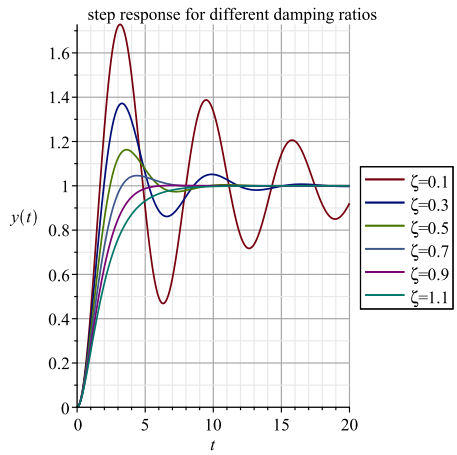
Maple

```
restart;
alias(DS=DynamicSystems):
H   := (w,zeta)->w^2/(s^2+2*zeta*w*s+w^2):
sys := (w,zeta)->DS:-TransferFunction(H(w,zeta)):
zetaValues := [seq(i,i=0.1..1.2,.2)]:
sol:=map(z->DS:-Simulate(sys(1,z),
                 Heaviside(t)),zetaValues):
c :=ColorTools:-GetPalette("Niagara"):

plots:-display(seq(plots[odeplot]
       (sol[i],t=0..20,color=c[i],

 legend=typeset(zeta,"=",zetaValues[i]),
 legendstyle=[location=right]),
         i=1..nops(zetaValues)),
         gridlines=true,
 title="step response for different damping ratios",
 labels=[typeset(t),typeset(y(t))]);
```

Instead of using Simulate as above, another option is to use `ResponsePlot` which gives same plot as above.

```
restart;
alias(DS=DynamicSystems):
c:=ColorTools:-GetPalette("Niagara"):
H:=(w,zeta)->w^2/(s^2+2*zeta*w*s+w^2):
sys:= (w,zeta)->
    DS:-TransferFunction(H(w,zeta)):
zetaValues := [seq(i,i=0.1..1.2,.2)]:

sol:=map(i->DS:-ResponsePlot(sys(1,zetaValues[i]),
 Heaviside(t),
 duration=14,
 color=c[i],
 legend=typeset(zeta,"=",zetaValues[i]),
 legendstyle=[location=right]
 ),
  [seq(i,i=1..nops(zetaValues))]
   ):
```

```
plots:-display(sol,gridlines=true,
 title="step response for different damping ratios",
 labels=[typeset(t),typeset(y(t))]
  );
```



1.12 View steady state error of 2nd order LTI system with changing undamped natural frequency

Problem: Given the transfer function

$$H(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

Display the output and input on the same plot showing how the steady state error changes as the un damped natural frequency ω_n changes. Do this for ramp and step input.

The steady state error is the difference between the input and output for large time. In other words, it the difference between the input and output at the time when the response settles down and stops changing.

Displaying the curve of the output and input on the same plot allows one to visually see steady state error.

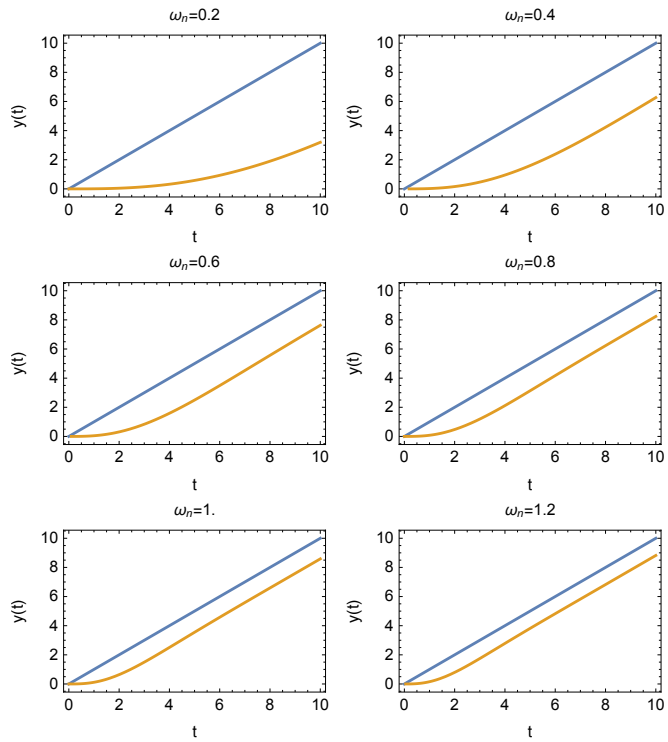
Use maximum time of 10 seconds and $\xi = 0.707$ and change ω_n from 0.2 to 1.2.

Do this for ramp input and for unit step input. It can be seen that with ramp input, the steady state do not become zero even at steady state. While with step input, the steady state error can become zero.

1.12.1 Mathematica

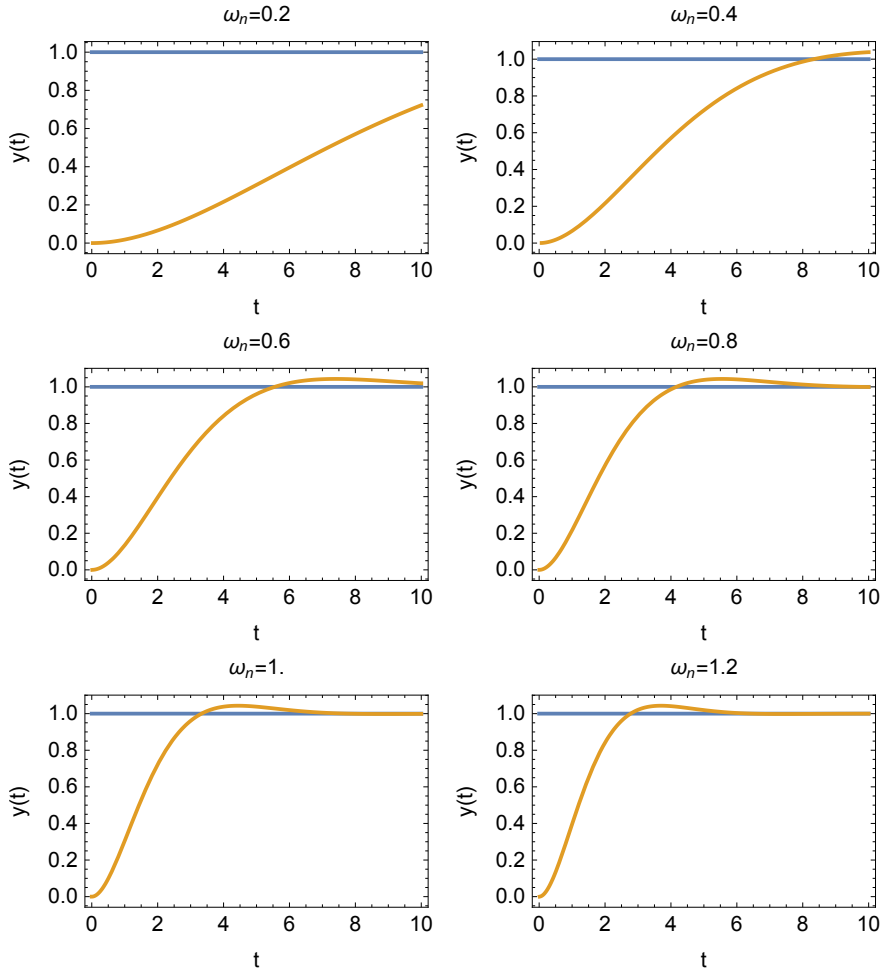
1.12.1.1 ramp input

```
Remove["Global`*"];
sys = TransferFunctionModel[w^2 /(s^2+2 z w  s+w^2 ),s];
z   = .707;
nTrials = 6;
maxTime = 10;

tb = Table[Plot[
  Evaluate@{t,OutputResponse[sys/.w->0.2*i,t,t]},
     {t,0,maxTime},
     Frame->True,
     PlotRange->All,
     FrameLabel->{
      {"y(t)",None},
      {"t","Subscript[\[Omega], n]="<>ToString[0.2*i]}
      }
    ],
    {i,1,nTrials}];
Grid[Partition[tb,2]]
```



1.12.1.2 step input

```
tb = Table[Plot[
  Evaluate@{UnitStep[t],
     OutputResponse[sys/.w->0.2*i,UnitStep[t],t]},
     {t,0,maxTime},
     Frame->True,
     PlotRange->All,
     FrameLabel->{
      {"y(t)",None},
      {"t","Subscript[\[Omega], n]="<>ToString[0.2*i]}
      }
    ],
    {i,1,nTrials}];

Grid[Partition[tb,2]]
```



1.13 Show the use of the inverse Z transform

These examples show how to use the inverse a Z transform.

1.13.1 example 1

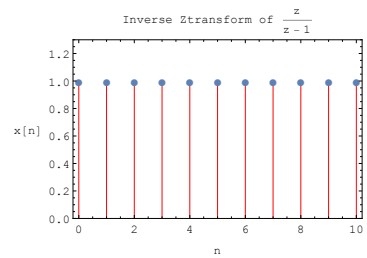
Problem: Given

$$F(z) = \frac{z}{z-1}$$

find $x[n] = F^{-1}(z)$ which is the inverse Ztransform.

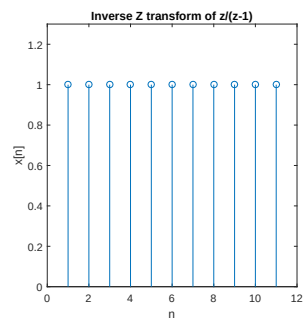
Mathematica

```
Clear["Global`*"];

x[n_]:= InverseZTransform[z/(z-1),z,n]

ListPlot[Table[{n,x[n]},{n,0,10}],
  Frame->True,
  FrameLabel->{{"x[n]",None},
      {"n","Inverse Ztransform of z/(z-1)"}},
  FormatType->StandardForm,
  RotateLabel->False,
  Filling->Axis,
  FillingStyle->Red,
  PlotRange->{Automatic,{0,1.3}},
  PlotMarkers->{Automatic,12}]
```



Matlab

```
function e19()

nSamples = 10;
z = tf('z');
h = z/(z-1);
[num,den] = tfdata(h,'v');
[delta,~] = impseq(0,0,nSamples);
xn = filter(num,den,delta);

stem(xn);
title('Inverse Z transform of z/(z-1)');
xlabel('n'); ylabel('x[n]');
ylim([0 1.3]);
set(gcf,'Position',[10,10,400,400]);

end

%function from Signal Processing by Proakis
%corrected a little by me for newer matlab version
function [x,n] = impseq(n0,n1,n2)
% Generates x(n) = delta(n-n0); n1 <= n,n0 <= n2
% ---------------------------------------------
% [x,n] = impseq(n0,n1,n2)
%
if ((n0 < n1) || (n0 > n2) || (n1 > n2))
    error('arguments must satisfy n1 <= n0 <= n2')
end
n = n1:n2;
x = (n-n0) == 0;
end
```



1.13.2 example 2

Problem: Given

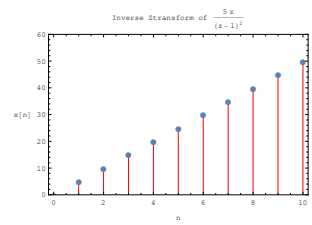
$$F(z) = \frac{5z}{(z-1)^2}$$

find $x[n] = F^{-1}(z)$

In Mathematica analytical expression of the inverse Z transform can be generated as well as shown below

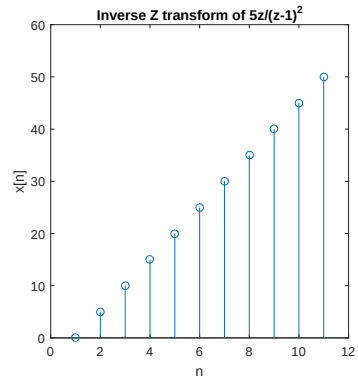
Mathematica

```
Clear["Global`*"];
x[n_]:= InverseZTransform[(5 z)/(z-1)^2,z,n];

ListPlot[Table[{n,x[n]},{n,0,10}],
   Frame->True,
   FrameLabel->{{"x[n]",None},
  {"n","Inverse Ztransform of (5 z)/(z-1)^2"}},
   FormatType->StandardForm,
   RotateLabel->False,
   Filling->Axis,
   FillingStyle->Red,
   PlotRange->{Automatic,{0,60}},
   PlotMarkers->{Automatic,12},
   ImageSize->350]
```



Matlab

```
function e19_2()

nSamples = 10;
z = tf('z');
h = (5*z)/(z-1)^2;
[num,den] = tfdata(h,'v');
[delta,~] = impseq(0,0,nSamples);
xn = filter(num,den,delta);

stem(xn);
title('Inverse Z transform of 5z/(z-1)^2');
xlabel('n'); ylabel('x[n]');
ylim([0 60]);
set(gcf,'Position',[10,10,400,400]);

end
```



1.14 Find the Z transform of sequence x(n)

1.14.1 example 1

Find the Z transform for the unit step discrete function

Given the unit step function $x[n] = u[n]$ defined as $x = \{1, 1, 1, \cdots\}$ for $n \geq 0$, find its Z transform.

Mathematica

```
Remove["Global`*"];
ZTransform[UnitStep[n],n,z]
```

```
Out[] = z/(-1+z)
```

Matlab

```
syms n
pretty(ztrans(heaviside(n)))
```

```
  1
----- + 1/2
z - 1
```

1.14.2 example 2

Find the Z transform for $x[n] = \left(\frac{1}{3}\right)^n u(n) + (0.9)^{n-3} u(n)$

Mathematica

```
f[n_]:=((1/3)^n+(9/10)^(n-3))UnitStep[n];
ZTransform[f[n],n,z]
```

```
z (3/(-1+3 z)+10000/(729 (-9+10 z)))
```

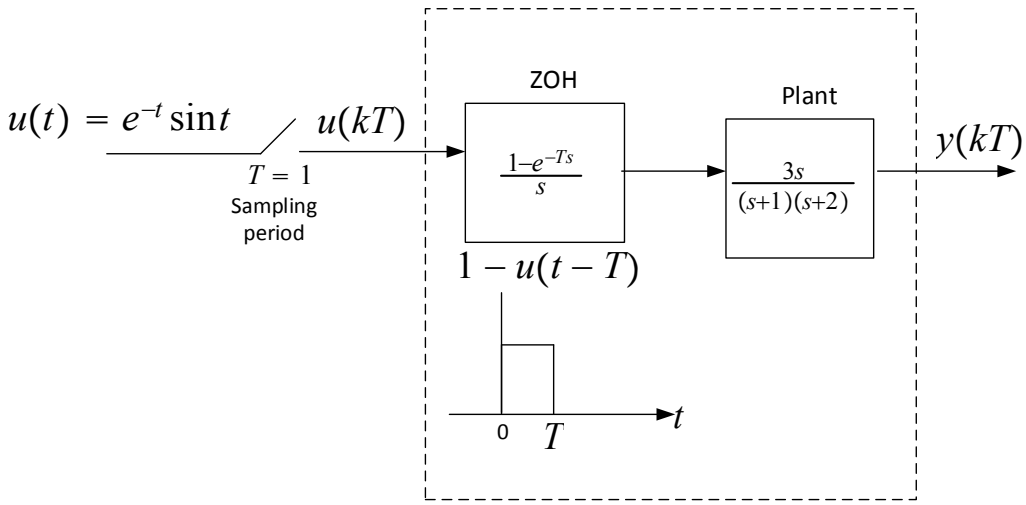
Matlab

```
syms n
pretty(ztrans(
 ((1/3)^n+(0.9)^(n-3))
       *heaviside(n)))
```

```
      100           1
------------- + ------- + 1729/1458
81 (z - 9/10)   3 z - 1
```

1.15 Sample a continuous time system

Given the following system, sample the input and find and plot the plant output



Equivalent discrete time system is given by the Z transform of the ZOH Laplace transform times the plant transfer function $Z\left(\frac{1-e^{-Ts}}{s}\frac{3s}{(s+1)(s+2)}\right)$

```
plant = TransferFunctionModel[3 s/((s + 1) (s + 2)), s]
ToDiscreteTimeModel[plant, sampleTime, z, Method → "ZeroOrderHold"]
```

In Mathematica, the above is implemented as follows

```
s = tf('s');
plant = (1/s)*(1/(s+0.5));
c2d(plant,sampleTime,'zoh')
```

In Matlab it is implemented as follows

Use sampling frequency $f = 1$ Hz and show the result for up to 14 seconds. Use as input the signal $u(t) = \exp(-0.3t)\sin(2\pi(f/3)t)$.

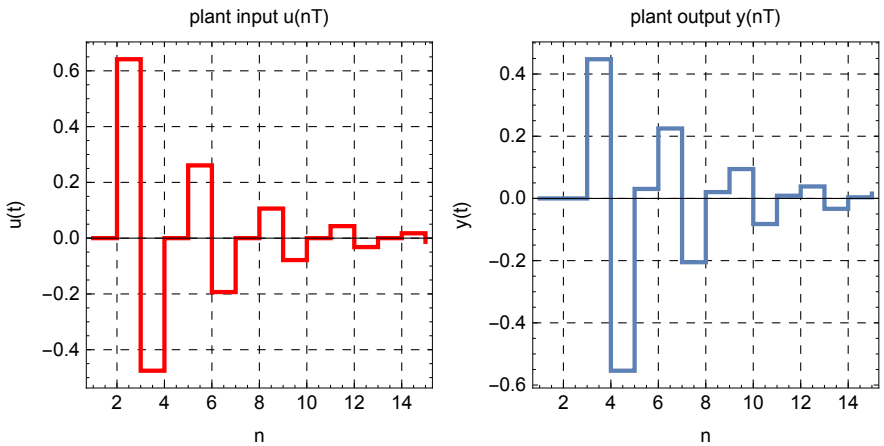
Plot the input and output on separate plots, and also plot them on the same plot.

Mathematica

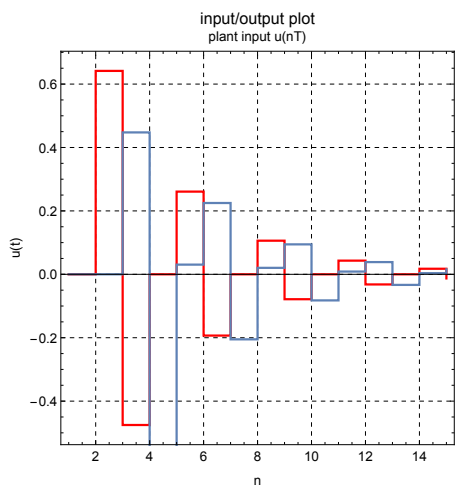
```
Clear["Global`*"];

nSamples =14;
f =1.0;
sampleTime =1/f;
u[t_] := Sin[2*Pi*(f/3)*t]*Exp[-0.3*t]
ud[n_]:= u[n*sampleTime];
plant = TransferFunctionModel[
            (3 s)/((s+1)(s+2)),s];
sysd  = ToDiscreteTimeModel[plant,
         sampleTime,
         z,
         Method->"ZeroOrderHold"]
```

$$\left(\frac{-0.697632 + 0.697632\,z}{0.0497871 - 0.503215\,z + z^2}\right)_{1.}^{\mathcal{T}}$$

```
opts = {Joined->True,PlotRange->All,
  AspectRatio->1,InterpolationOrder->0,
  ImageSize->200,Frame->True,
  PlotRange->{{0,nSamples},{-0.5,0.5}},
  GridLines->Automatic,GridLinesStyle->Dashed};

inputPlot = ListPlot[Table[ud[k],{k,0,nSamples}],
   Evaluate[opts],
   FrameLabel->{{"u(t)",None},
             {"n","plant input u(nT)"}},
   PlotStyle->{Thick,Red}];

plantPlot=ListPlot[OutputResponse[sysd,Table[ud[k],
 {k,0,nSamples}]],
 Evaluate[opts],
 FrameLabel->{{"y(t)",None},
             {"n","plant output y(nT)"}},
 PlotStyle->{Thick,Blue}];

Grid[{{inputPlot,plantPlot}}]
```




```
Show[{inputPlot,plantPlot},
     PlotLabel->"input/output plot"]
```

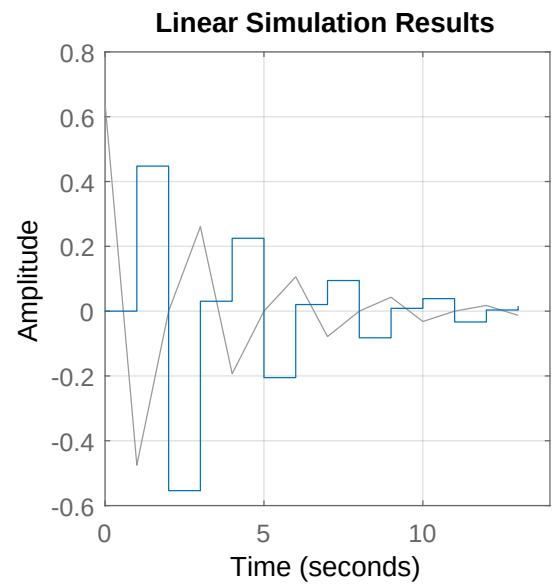


Matlab

```
clear all; close all;

nSamples = 14;
f = 1;
T = 1/f;
u=@(t) sin(2*pi*(f/3).*t).*exp(-0.3.*t);
ud=@(n) u(n*T);
U=ud(1:14);  %sampled input
s=tf('s');
plant  = (3*s)/((s+1)*(s+2));
plantD = c2d(plant,T,'zoh')
```

```
plantD =

     0.6976 z - 0.6976
  ------------------------
  z^2 - 0.5032 z + 0.04979

Sample time: 1 seconds
Discrete-time transfer function.
```

```
lsim(plantD,U,0:nSamples-1)
grid
```

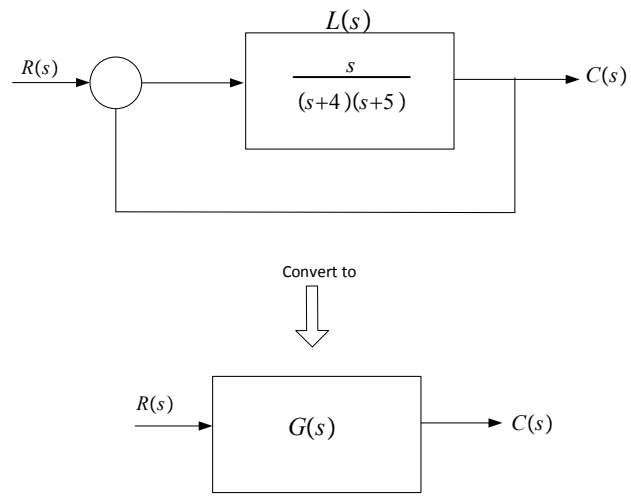


1.16 Find closed loop transfer function from the open loop transfer function for a unity feedback

Problem: Given

$$L(s) = \frac{s}{(s+4)(s+5)}$$

as the open loop transfer function, how to find $G(s)$, the closed loop transfer function for a unity feedback?



Mathematica

```
Clear["Global`*"];
sys = TransferFunctionModel[
            s/((s+4)(s+5)),s]
```

$$\left(\frac{s}{(4+s)\,(5+s)}\right)_{\mathcal{T}}$$

```
closedLoop = SystemsModelFeedbackConnect[sys];
```

$$\left(\frac{s}{20+10\,s+s^2}\right)_{\mathcal{T}}$$

The system wrapper can be removed in order to obtain the rational polynomial expression as follows

```
First[Flatten[closedLoop[[1,1]]/
        closedLoop[[1,2]]]]
```

$$\frac{s}{s^2+10s+20}$$

Matlab

```
clear all; close all;
s=tf('s');
sys=s/((s+4)*(s+5));
closedloop=feedback(sys,1)
```

```
Transfer function:
       s
---------------
s^2 + 10 s + 20
```

1.17 Compute the Jordan canonical/normal form of a matrix A

Mathematica

```
Remove["Global`*"];
m={{3,-1,1,1,0,0},
   {1, 1,-1,-1,0,0},
  {0,0,2,0,1,1},
  {0,0,0,2,-1,-1},
  {0,0,0,0,1,1},
  {0,0,0,0,1,1}};
MatrixForm[m]
```

$$\begin{pmatrix} 3 & -1 & 1 & 1 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

```
{a,b}=JordanDecomposition[m];
b
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

Matlab

```
clear all;
A=[3 -1  1  1  0  0;
   1  1 -1 -1  0  0;
   0  0  2  0  1  1;
   0  0  0  2 -1 -1;
   0  0  0  0  1  1;
   0  0  0  0  1  1;]

jordan(A)
```

```
ans =
 0     0     0     0     0     0
 0     2     1     0     0     0
 0     0     2     1     0     0
 0     0     0     2     0     0
 0     0     0     0     2     1
 0     0     0     0     0     2
```

Maple

```
restart;
A:=Matrix([[3,-1,1,1,0,0],
   [1, 1,-1,-1,0,0],
   [0,0,2,0,1,1],
   [0,0,0,2,-1,-1],
   [0,0,0,0,1,1],
   [0,0,0,0,1,1]]);
LinearAlgebra:-JordanForm(A);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

1.18 Solve the continuous-time algebraic Riccati equation

Problem: Solve for X in the Riccati equation

$$A'X + XA - XBR^{-1}B'X + C'C = 0$$

given

$$A = \begin{pmatrix} -3 & 2 \\ 1 & 1 \end{pmatrix}$$
$$B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
$$C = \begin{pmatrix} 1 & -1 \end{pmatrix}$$
$$R = 3$$

Mathematica

```
Clear ["Global`*"];
a={{-3,2},{1,1}};
b={{0},{1}};
c={{1,-1}};
r={{3}};
sol=RiccatiSolve[{a,b},{Transpose[c].c,r}];
MatrixForm[N[sol]]
```

$$\begin{pmatrix} 0.589517 & 1.82157 \\ 1.82157 & 8.81884 \end{pmatrix}$$

Matlab

```
%needs control system
clear all; close all;
a = [-3 2;1 1];
b = [0 ; 1];
c = [1 -1];
r = 3;
x = care(a,b,c'*c,r)
```

```
x =
    0.5895    1.8216
    1.8216    8.8188
```

Maple

```
restart;
A:=Matrix([[-3,2],[1,1]]);
B:=Vector([0,1]);
C:=Vector[row]([1,-1]);
Q:=C^%T.C;
R:=Matrix([[3]]);
LinearAlgebra:-CARE(A,B,Q,R)
```

$$\begin{bmatrix} 0.5895174373 & 1.8215747249 \\ 1.8215747249 & 8.8188398069 \end{bmatrix}$$

1.19 Solve the discrete-time algebraic Riccati equation

Problem: Given a continuous-time system represented by a transfer function

$$\frac{1}{s(s+0.5)}$$

convert this representation to state space and sample the system at sampling period of 1 second, and then solve the discrete-time Riccati equation.

The Riccati equation is given by

$$A'X + XA - XBR^{-1}B'X + C'C = 0$$

Let $R = [3]$.

Mathematica

```
Clear["Global`*"];
sys=TransferFunctionModel[1/(s(s+.5)),s];
dsys=ToDiscreteTimeModel[sys,
        1,z,Method->"ZeroOrderHold"]
```

$$\left(\frac{0.360816 + 0.426123\,z}{0.606531 - 1.60653\,z + z^2}\right)_{1.}^{\mathcal{T}}$$

```
ss = StateSpaceModel[dsys]
```

$$\left(\begin{array}{cc|c} 0 & 1. & 0 \\ \hline -0.606531 & 1.60653 & 1. \\ \hline 0.360816 & 0.426123 & 0 \end{array}\right)_{1.}^{\mathcal{S}}$$

```
a = ss[[1,1]];
b = ss[[1,2]];
c = ss[[1,3]];
d = ss[[1,4]];
r = {{3}};
DiscreteRiccatiSolve[{a,b},
               {Transpose[c].c,r}];
MatrixForm[%]
```

$$\begin{pmatrix} 0.671414 & -0.977632 \\ -0.977632 & 2.88699 \end{pmatrix}$$

Matlab

```
clear all; close all;
s    = tf('s');
sys  = 1/(s*(s+0.5));
dsys = c2d(sys,1)
```

```
dsys =
    0.4261 z + 0.3608
  ----------------------
  z^2 - 1.607 z + 0.6065

Sample time: 1 seconds
Discrete-time transfer function.
```

```
[A,B,C,D]=dssdata(dsys)
```

```
A =
    1.6065   -0.6065
    1.0000         0
B =
     1
     0
C =
    0.4261    0.3608
D =
     0

    2.8870   -0.9776
   -0.9776    0.6714
```

```
dare(A,B,C'*C,3)
```

```
ans =

    2.8870   -0.9776
   -0.9776    0.6714
```

Maple

```
restart;
alias(DS=DynamicSystems):
sys := DS:-TransferFunction(1/(s*(s+1/2)));
sys := DS:-ToDiscrete(sys, 1, 'method'='zoh');
sys := DS:-StateSpace(sys);
Q:=sys:-c^%T.sys:-c;
R:=Matrix([[3]]);
LinearAlgebra:-DARE(sys:-a,sys:-b,Q,R)
```

$$\begin{bmatrix} 0.6714144604 & -0.9776322436 \\ -0.9776322436 & 2.8869912178 \end{bmatrix}$$

1.20 Display impulse response of H(z) and the impulse response of its continuous time approximation H(s)

Plot the impulse response of $H(z) = z/(z^2 - 1.4z + 0.5)$ and using sampling period of $T = 0.5$ find continuous time approximation using zero order hold and show the impulse response of the system and compare both responses.

Mathematica

```
maxSimulationTime = 10;
samplePeriod      = 0.5;
tf = TransferFunctionModel[z/(z^2-1.4 z+0.5),
          z,
          SamplingPeriod->samplePeriod];
```

$$\left(\frac{z}{0.5 - 1.4\,z + z^2}\right)_{0.5}^{\mathcal{T}}$$

Find its impulse response

```
discreteResponse=First@OutputResponse[tf,DiscreteDelta[k],
                                        {k,0,maxSimulationTime}]
```

```
{0.,1.,1.4,1.46,1.344,1.1516,0.94024,0.740536,
0.56663,0.423015,0.308905,0.22096,0.154891,
```

```
0.106368,0.0714694,0.0468732,0.0298878,
0.0184063,0.0108249,0.00595175,0.00291999}
```

approximate to continuous time, use ZeroOrderHold

```
ctf = ToContinuousTimeModel[tf, s, Method -> "ZeroOrderHold"]
```

$$\left(\frac{5.60992 + 1.25559\,s}{0.560992 + 1.38629\,s + s^2}\right)^{\mathcal{T}}$$

Find the impulse response of the continuous time system

```
continouseTimeResponse=Chop@First@OutputResponse[ctf,DiracDelta[t],t]
```

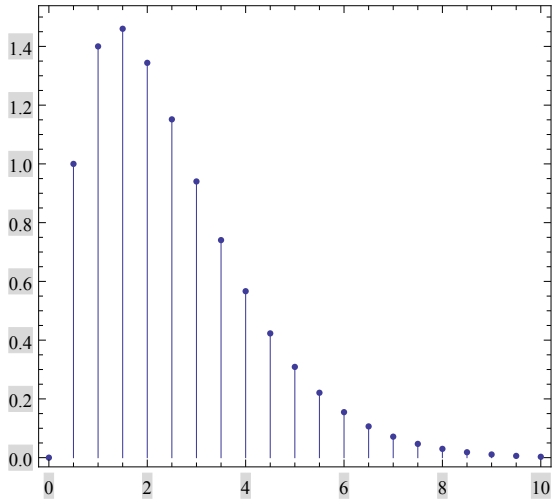
$$-1.25559e^{-0.693147t}(-13.3012\theta(t)\sin(0.283794t) - 1.\theta(t)\cos(0.283794t))$$

```
continouseTimeResponse=Chop@First@OutputResponse[ctf,DiracDelta[t],
                                                 {t,0,maxSimulationTime}]
```

InterpolatingFunction[ Domain: {{0., 10.}} Output: scalar][t]

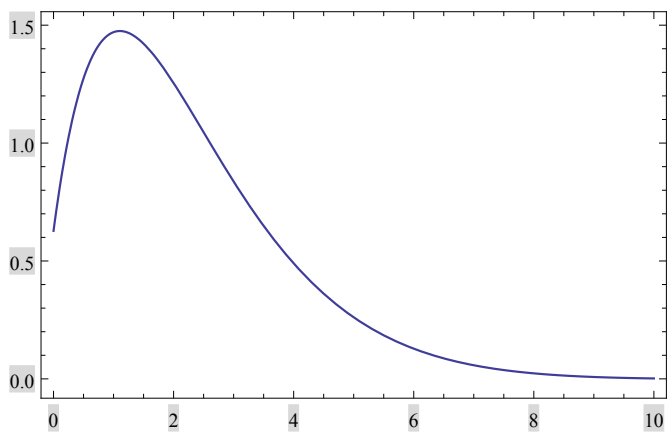
Plot the impulse response of the discrete system

```
ListPlot[
discreteResponse,
DataRange->{0,maxSimulationTime},
Filling->Axis,
FillingStyle->{Red,
  PlotMarkers->Graphics[{PointSize[0.03],
  Point[{0,0}]}]},PlotRange->All,
  ImageSize->300,ImagePadding->{{45,10},{35,10}},
  AspectRatio->0.9,AxesOrigin->{0,0},Frame->True,Axes->None,
  ImageSize->300,
  Frame->True
  ]
```



Plot the impulse response of the continuous system

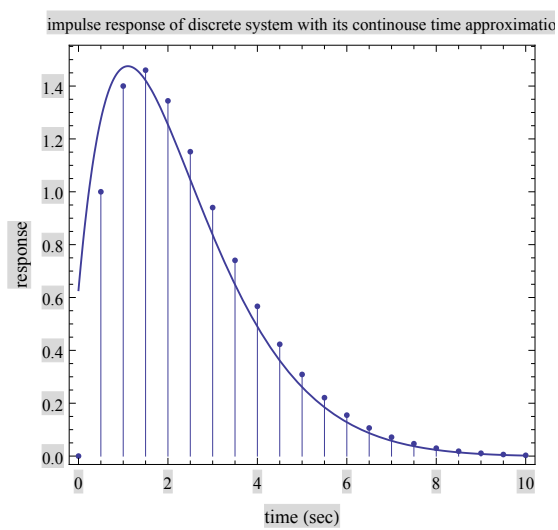
```
Plot[samplePeriod *continouseTimeResponse,{t,0,maxSimulationTime},
     PlotRange->All,Frame->True,ImageSize->300]
```



Plot both responses on the same plot

```
p = Show[
  ListPlot[
   discreteResponse,
   Filling -> Axis,
   FillingStyle -> {Red,
     PlotMarkers -> Graphics[{PointSize[0.03], Point[{0, 0}]}]}],
   PlotRange -> All,
   DataRange -> {0, maxSimulationTime},
```

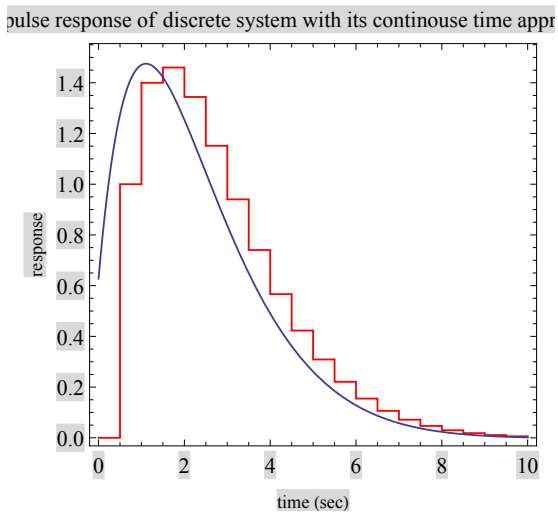
```
   ImageSize -> 300,
   ImagePadding -> {{45, 10}, {35, 50}},
   AspectRatio -> 0.9,
   AxesOrigin -> {0, 0},
   Frame -> True,
   Axes -> None],
  Plot[samplePeriod *continouseTimeResponse, {t, 0, maxSimulationTime},
   PlotRange -> All],
  PlotRange -> All,
  FrameLabel -> {{Style["response", 10], None},
     {Style["time (sec)", 10],
    "impulse response of discrete system \
     with its continouse time approximatio"}}
  ]
```



Do the same plot above, using stair case approximation for the discrete plot

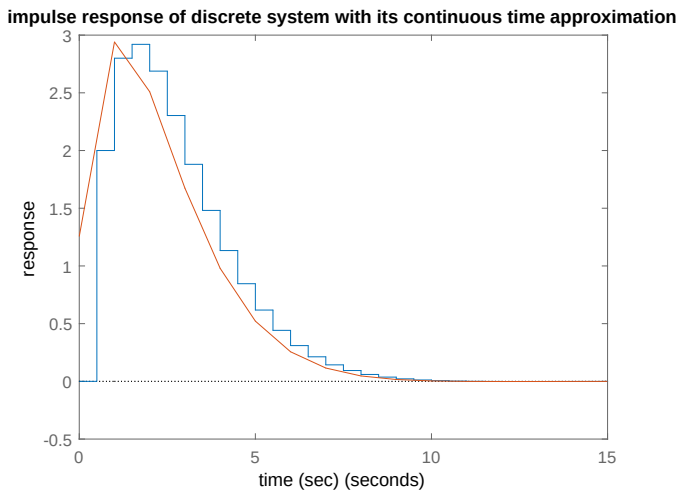
```
Show[
ListPlot[
 discreteResponse,
 Filling->None,PlotRange->All,DataRange->{0,maxSimulationTime},
 ImageSize->300,ImagePadding->{{45,10},{35,50}},AspectRatio->0.9,
 AxesOrigin->{0,0},Frame->True,Axes->None,
 Joined->True,InterpolationOrder->0,PlotStyle->Red],
Plot[samplePeriod *continouseTimeResponse,{t,0,maxSimulationTime},
 PlotRange->All],
 PlotRange->All,FrameLabel->{{Style["response",10],None},
```

```
{Style["time (sec)",10],"impulse response"}}]
```



Matlab

```
clear all;
maxSimulationTime = 15;
samplePeriod = 0.5;

z=tf('z',samplePeriod);
tf=z/(z^2-1.4*z+0.5)
impulse(tf,0:samplePeriod:maxSimulationTime)

ctf=d2c(tf,'zoh')

hold on
impulse(ctf,0:maxSimulationTime)
title(['impulse response of discrete system with',...
       ' its continuous time approximation']);
xlabel('time (sec)');
ylabel('response');
```



1.21 Find the system type given an open loop transfer function

Problem: Find the system type for the following transfer functions

1. $\frac{s+1}{s^2-s}$
2. $\frac{s+1}{s^3-s^2}$
3. $\frac{s+1}{s^5}$

To find the system type, the transfer function is put in the form $\frac{k\sum_i(s-s_i)}{s^M\sum_j(s-s_j)}$, then the system type is the exponent M. Hence it can be seen that the first system above has type one since the denominator can be written as $s^1(s-1)$ and the second system has type 2 since the denominator can be written as $s^2(s-1)$ and the third system has type 5. The following computation determines the type

Mathematica

```
Clear["Global`*"];
p=TransferFunctionPoles[TransferFunctionModel[
    (s+1)/(s^2-s),s]];
Count[Flatten[p],0]
```

Out[171]= 1

```
p=TransferFunctionPoles[TransferFunctionModel[
    (s+1)/( s^3-s^2),s]];
Count[Flatten[p],0]
```

Out[173]= 2

```
p=TransferFunctionPoles[
   TransferFunctionModel[(s+1)/ s^5 ,s]];
Count[Flatten[p],0]
```

Out[175]= 5

Matlab

```
clear all;
s=tf('s');
[~,p,~]=zpkdata((s+1)/(s^2-s));
length(find(p{:}==0))
```

```
ans =
     1
```

```
[~,p,~]=zpkdata((s+1)/(s^3-s^2));
length(find(p{:}==0))
```

```
ans =
     2
```

```
[~,p,~]=zpkdata((s+1)/s^5);
length(find(p{:}==0))
```

```
ans =
     5
```

1.22 Find the eigenvalues and eigenvectors of a matrix

Problem, given the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Find its eigenvalues and eigenvectors.

Mathematica

```
Remove["Global`*"]
(a = {{1,2,3}, {4,5,6}, {7,8,9}})
   // MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
Eigenvalues[a]
N[%]
```

$$\left\{\frac{3}{2}\left(5+\sqrt{33}\right), \frac{3}{2}\left(5-\sqrt{33}\right), 0\right\}$$

$$\{16.1168, -1.11684, 0.\}$$

```
Eigenvectors[a]
N[%]
```

$$\begin{pmatrix} -\frac{-15-\sqrt{33}}{33+7\sqrt{33}} & \frac{4\left(6+\sqrt{33}\right)}{33+7\sqrt{33}} & 1 \\ -\frac{15-\sqrt{33}}{-33+7\sqrt{33}} & \frac{4\left(-6+\sqrt{33}\right)}{-33+7\sqrt{33}} & 1 \\ 1 & -2 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0.283349 & 0.641675 & 1. \\ -1.28335 & -0.141675 & 1. \\ 1. & -2. & 1. \end{pmatrix}$$

Matlab

Matlab generated eigenvectors are such that the sum of the squares of the eigenvector elements add to one.

```
clear all; close all;

A=[1 2 3;4 5 6;7 8 9];

[v,e]=eig(A)
```

```
v =
   -0.2320   -0.7858    0.4082
   -0.5253   -0.0868   -0.8165
   -0.8187    0.6123    0.4082

e =
   16.1168         0         0
         0   -1.1168         0
         0         0   -0.0000
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
evalf(LinearAlgebra:-Eigenvectors(A))
```

First vector shows eigenvalues, and matrix on the right shows the eigenvectors in same order.

$$\left[\begin{bmatrix} 16.1168439700 \\ -1.1168439700 \\ 0.0 \end{bmatrix}, \begin{bmatrix} 0.2833494518 & -1.2833494520 & 1.0 \\ 0.6416747260 & -0.1416747258 & -2.0 \\ 1.0 & 1.0 & 1.0 \end{bmatrix}\right]$$

1.23 Find the characteristic polynomial of a matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}$$

Mathematica

```
a = {{1,2,3},{4,5,6},{7,8,0}};
CharacteristicPolynomial[a,x]
```

$$-x^3 + 6x^2 + 72x + 27$$

Matlab

Note: Matlab and Maple generated characteristic polynomial coefficients are negative to what Mathematica generated.

But the sign difference is not important.

```
clear all;
A=[1 2 3;4 5 6;7 8 0];

p=poly(A)
poly2str(p,'x')
```

```
p =
 1.0000 -6.0000 -72.0000 -27.0000
ans =
 x^3 - 6 x^2 - 72 x - 27
```

Maple

```
restart;
A:=Matrix([[1,2,3],[4,5,6],[7,8,0]]);
LinearAlgebra:-CharacteristicPolynomial(A,x)
```

$$x^3 - 6x^2 - 72x - 27$$

1.24 Verify the Cayley-Hamilton theorem that every matrix is zero of its characteristic polynomial

Problem, given the matrix

$$\begin{pmatrix} 1 & 2 \\ 3 & 2 \end{pmatrix}$$

Verify that matrix is a zero of its characteristic polynomial. The Characteristic polynomial of the matrix is found, then evaluated for the matrix. The result should be the zero matrix.

Mathematica

```
Remove["Global`*"]
a = {{1,2},{3,2}};
n = Length[a];
p = CharacteristicPolynomial[a,x]
```

$$x^2 - 3x - 4$$

```
(-4 IdentityMatrix[n] - 3 a +
  MatrixPower[a,2])//MatrixForm
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Another way is as follows

```
a  = {{1,2},{3,2}};
p  = CharacteristicPolynomial[a,x];
cl = CoefficientList[p,x];
Sum[MatrixPower[a,j-1] cl[[j]],
     {j,1,Length[cl]}]
```

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Matlab

MATLAB has a build-in function polyvalm() to do this more easily than in Mathematica. Although the method shown in Mathematica can easily be made into a Matlab function

```
clear;
A=[1 2;3 2];
p=poly(A);
poly2str(p,'x')
polyvalm(p,A)
```

```
ans =
   x^2 - 3 x - 4
ans =
     0     0
     0     0
```

1.25 How to check for stability of system represented as a transfer function and state space

Problem: Given a system Laplace transfer function, check if it is stable, then convert to state space and check stability again. In transfer function representation, the check is that all poles of the transfer function (or the zeros of the denominator) have negative real part. In state space, the check is that the matrix A is negative definite. This is done by checking that all the eigenvalues of the matrix A have negative real part. The poles of the transfer function are the same as the eigenvalues of the A matrix. Use

$$sys = \frac{5s}{s^2 + 4s + 25}$$

1.25.1 Checking stability using transfer function poles

Mathematica

```
Remove["Global`*"];
sys  =TransferFunctionModel[(5s)/(s^2+4s+25),s];
poles=TransferFunctionPoles[sys]
```

```
{{{-2-I Sqrt[21],-2+I Sqrt[21]}}}
```

```
Re[#]&/@poles
```

```
Out[42]= {{{-2,-2}}}
```

```
Select[%,#>=0&]
```

```
Out[44]= {}
```

Matlab

```
clear all;
s=tf('s');
sys=5*s/( s^2+4*s+25);
[z,p,k]=zpkdata(sys,'v');
p
```

```
>>
p =
  -2.0000 + 4.5826i
  -2.0000 - 4.5826i
```

```
find(real(p)>=0)
```

```
ans =
   Empty matrix: 0-by-1
```

1.25.2 Checking stability using state space A matrix

Mathematica

```
ss=StateSpaceModel[sys];
a=ss[[1,1]]
```

```
Out[49]= {{0,1},{-25,-4}}
```

```
e=Eigenvalues[a]
```

```
Out[50]= {-2+I Sqrt[21],-2-I Sqrt[21]}
```

```
Re[#]&/@e
```

```
Out[51]= {-2,-2}
```

```
Select[%,#>=0&]
```

```
Out[52]= {}
```

Matlab

```
sys=ss(sys);
[A,B,C,D]=ssdata(sys);
A
```

```
A =
   -4.0000   -6.2500
    4.0000         0
```

```
e=eig(A)
```

```
e =
  -2.0000 + 4.5826i
  -2.0000 - 4.5826i
```

```
find(real(e)>=0)
```

```
ans =
   Empty matrix: 0-by-1
```

1.26 Check continuous system stability in the Lyapunov sense

Problem: Check the stability (in Lyapunov sense) for the state coefficient matrix

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{bmatrix}$$

The Lyapunov equation is solved using lyap() function in MATLAB and LyapunovSolve[] function in Mathematica, and then the solution is checked to be positive definite (i.e. all its eigenvalues are positive).

We must transpose the matrix A when calling these functions, since the Lyapunov equation is defined as $A^T P + PA = -Q$ and this is not how the software above defines them. By simply transposing the A matrix when calling them, then the result will be correct.

Mathematica

```
Remove["Global`*"];
(mat = {{0,1,0},{0,0,1},{-1,-2,-3}})
```

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{pmatrix}$$

```
p = LyapunovSolve[Transpose@mat,
    -IdentityMatrix[Length[mat]]];
MatrixForm[N[p]]
```

$$\begin{pmatrix} 2.3 & 2.1 & 0.5 \\ 2.1 & 4.6 & 1.3 \\ 0.5 & 1.3 & 0.6 \end{pmatrix}$$

```
N[Eigenvalues[p]]
```

$$\{6.18272, 1.1149, 0.202375\}$$

Matlab

```
clear all;

A=[0   1   0
   0   0   1
   -1 -2  -3];

p=lyap(A.',eye(length(A)))
```

```
p =
  2.3   2.1  0.5
  2.1   4.6  1.3
  0.5   1.3  0.6
```

```
e=eig(p)
```

```
e =
     0.20238
       1.1149
       6.1827
```

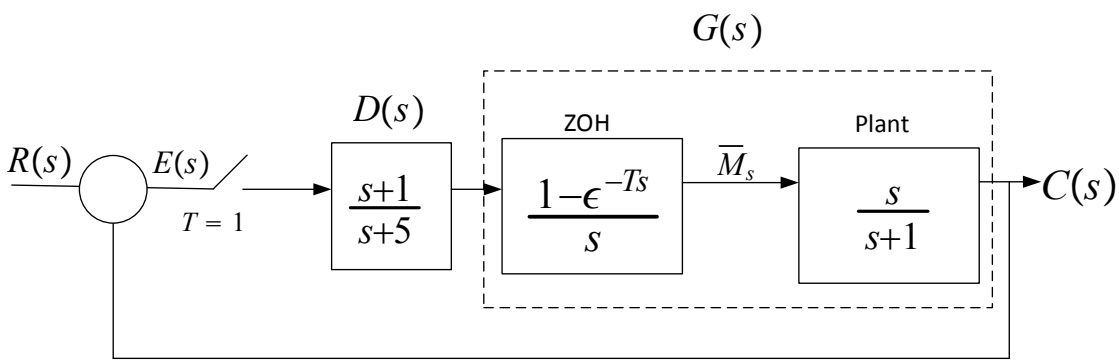
Maple

```
with(LinearAlgebra):
A:=<<0,1,0;0,0,1;-1,-2,-3>>;
p,s:=LyapunovSolve(A^%T,-<<1,0,0;0,1,0;0,0,1>>);
Eigenvalues(p);
```

$$\begin{bmatrix} 6.18272045921436 + 0.0\,i \\ 1.11490451203192 + 0.0\,i \\ 0.202375028753723 + 0.0\,i \end{bmatrix}$$

1.27 Given a closed loop block diagram, generate the closed loop Z transform and check its stability

Problem: Given the following block diagram, with sampling time $T = 0.1$ *sec*, generate the closed loop transfer function, and that poles of the closed loop transfer function are inside the unit circle



System block diagram.

Mathematica

```
Remove["Global`*"]
plant  = TransferFunctionModel[s/(1.0+s),s];
plantd = ToDiscreteTimeModel[
             plant,
             1,
             z,
             Method->"ZeroOrderHold"]
```

$$\left(\frac{1.-1.\,z}{0.367879-1.\,z}\right)^{\mathcal{T}}_{1.}$$

```
d = ToDiscreteTimeModel[
     TransferFunctionModel[(s+1)/(s+5.0),s],
          1,
          z,
          Method->"ZeroOrderHold"]
```

$$\left(\frac{0.801348-1.\,z}{0.00673795-1.\,z}\right)^{\mathcal{T}}_{1.}$$

```
sys=SystemsModelSeriesConnect[d,plantd]
```

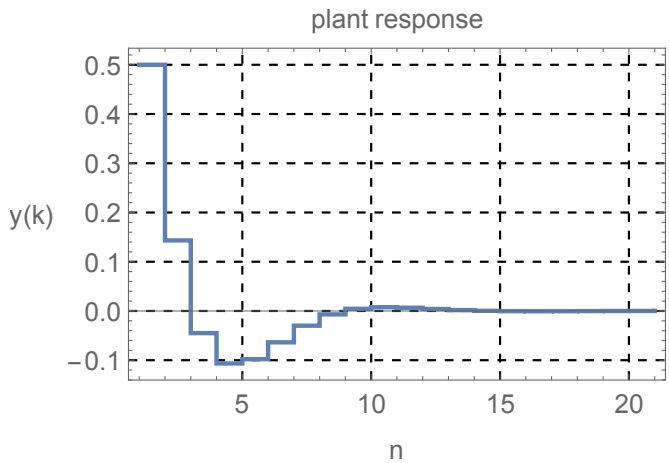
$$\left(\frac{(0.801348-1.\,z)\,(1.-1.\,z)}{(0.00673795-1.\,z)\,(0.367879-1.\,z)}\right)^{\mathcal{T}}_{1.}$$

```
loopBack=SystemsModelFeedbackConnect[sys]
```

$$\left(\frac{1.5825\times10^{17}\,(-1.+1.\,z)\,(-0.801348+1.\,z)}{3.92264\times10^{14}-5.92834\times10^{16}\,z+1.5825\times10^{17}\,z^2+1.5825\times10^{17}\,(-1.+1.\,z)\,(-0.801348+1.\,z)}\right)^{\mathcal{T}}_{1.}$$

Now generate unit step response

```
ListPlot[OutputResponse[loopBack,
      Table[1, {k, 0, 20}]],
        Joined->True,
        PlotRange->All,
        InterpolationOrder->0,
        Frame->True,
        PlotStyle->{Thick,Red},
        GridLines->Automatic,
        GridLinesStyle->Dashed,
        FrameLabel->{{"y(k)",None},
          {"n","plant response"}},
        RotateLabel->False]
```




```
ss=StateSpaceModel[loopBack]
```

$$\left(\begin{array}{cc|c} 0 & 1. & 0 \\ -0.401913 & 1.08798 & 1. \\ \hline 0.199717 & -0.356683 & 0.5 \end{array}\right)_{1.}^{S}$$

```
poles=TransferFunctionPoles[loopBack]
```

$$\{0.543991 - 0.325556i, 0.543991 + 0.325556i\}$$

```
Abs[#]&/@poles
```

$$\left(\{0.633966, 0.633966\} \right)$$

Poles are inside the unit circle, hence stable.

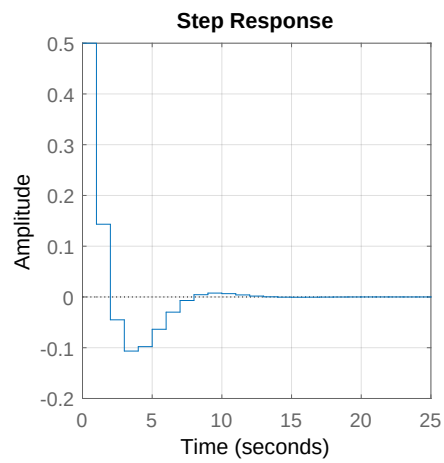
Matlab

```
clear all; close all

s      = tf('s');
plant  = s/(s+1);
T      = 1; %sampeling time;
plantd = c2d(plant,T,'zoh');
d      = c2d( (s+1)/(s+5), T , 'zoh');
```

```
% obtain the open loop
sys    = series(d,plantd);
% obtain the closed loop
loop   = feedback(sys,1)
```

```
loop =

   z^2 - 1.801 z + 0.8013
  ------------------------
  2 z^2 - 2.176 z + 0.8038

Sample time: 1 seconds
Discrete-time transfer function.
```

```
step(loop)
grid
```




```
[z,p,k]=zpkdata(loop,'v');
fprintf('poles of closed loop discrete system as inside unit circle\n');
abs(p)
```

```
ans =
    0.6340
    0.6340
```

1.28 Determine the state response of a system to only initial conditions in state space

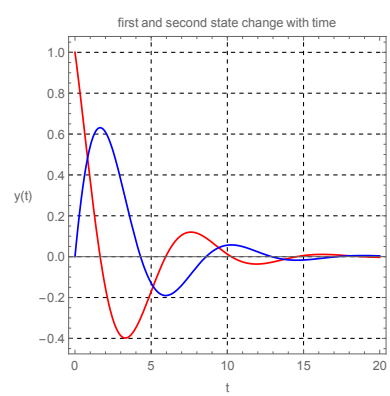
Problem: Given a system with 2 states, represented in state space, how to determine the state change due some existing initial conditions, when there is no input forces?

Mathematica

```
Remove["Global`*"];
a = {{-0.5572,-0.7814},{0.7814, 0}};
c = {{1.9691, 6.4493}};
sys=StateSpaceModel[
       {a,{{1},{0}},c,{{0}}}]
```

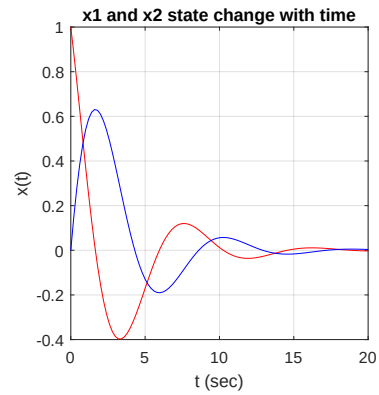
$$\left(\begin{array}{cc|c} -0.5572 & -0.7814 & 1 \\ 0.7814 & 0 & 0 \\ \hline 1.9691 & 6.4493 & 0 \end{array}\right) \boxed{S}$$

```
x0 = {1,0};
{x1,x2} = StateResponse[
            {sys,x0},0,{t,0,20}];

Plot[{x1,x2},{t,0,20},
  PlotRange->All,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  Frame->True,
  ImageSize->350,
  AspectRatio->1,
  FrameLabel->{{"y(t)",None},
   {"t",
   "first and second state\
    change with time"
   }},
    RotateLabel->False,
    PlotStyle->{Red,Blue},
    BaseStyle -> 12]
```



Matlab

```
clear;
A = [-0.5572   -0.7814;0.7814  0];
B = [1;0];
C = [1.9691  6.4493];
x0 = [1 ; 0];

sys = ss(A,B,C,[]);

[y,t,x]=initial(sys,x0,20);
plot(t,x(:,1),'r',t,x(:,2),'b');
title('x1 and x2 state change with time');
xlabel('t (sec)');
ylabel('x(t)');
grid
set(gcf,'Position',[10,10,320,320]);
```



1.29 Determine the response of a system to only initial conditions in state space

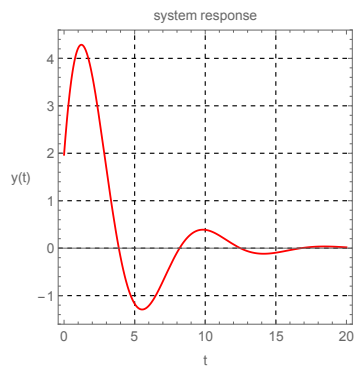
Problem: Given a system represented by state space, how to determine the response $y(t)$ due some existing initial conditions in the states. There is no input forces.

Mathematica

```
Remove["Global`*"];
a = {{-0.5572, -0.7814}, {0.7814, 0}};
c = {{1.9691, 6.4493}};
sys=StateSpaceModel[{a,{{1},{0}},c,{{0}}}]
```

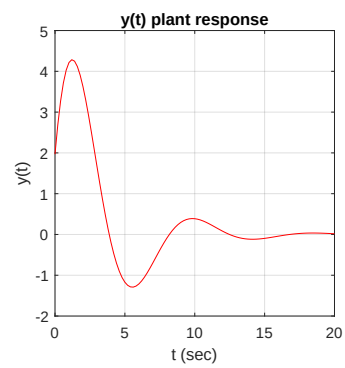
$$\left(\begin{array}{cc|c} -0.5572 & -0.7814 & 1 \\ \hline 0.7814 & 0 & 0 \\ \hline 1.9691 & 6.4493 & 0 \end{array}\right) \mathcal{S}$$

```
x0 = {1,0};  (*initial state vector*)
y  = OutputResponse[{sys,x0},0,{t,0,20}];
Plot[y,{t,0,20},
  PlotRange->All,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  Frame->True,
  ImageSize->300,
  AspectRatio->1,
  FrameLabel->{{"y(t)",None},
     {"t","system response"}},
  RotateLabel->False,PlotStyle->Red]
```



Matlab

```
clear all;

A = [-0.5572  -0.7814;
     0.7814  0];
B = [1;0];
C = [1.9691  6.4493];
x0 = [1 ; 0];

sys = ss(A,B,C,[]);

[y,t,x]=initial(sys,x0,20);
plot(t,y);
title('y(t) plant response');
xlabel('t (sec)');
ylabel('y(t)');
grid
set(gcf,'Position',...
   [10,10,320,320]);
```



1.30 Determine the response of a system to step input with nonzero initial conditions

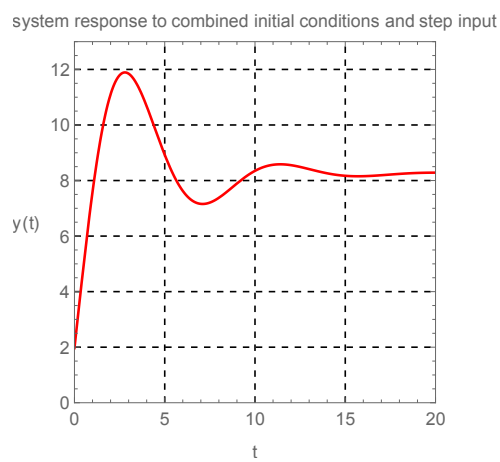
Problem: Given a system represented by state space, how to determine the response with nonzero initial conditions in the states and when the input is a step input?

Mathematica

```
Remove["Global`*"];
a = {{-0.5572, -0.7814}, {0.7814, 0}};
c = {{1.9691, 6.4493}};
sys=StateSpaceModel[
       {a,{{1},{0}},c,{{0}}}]
```

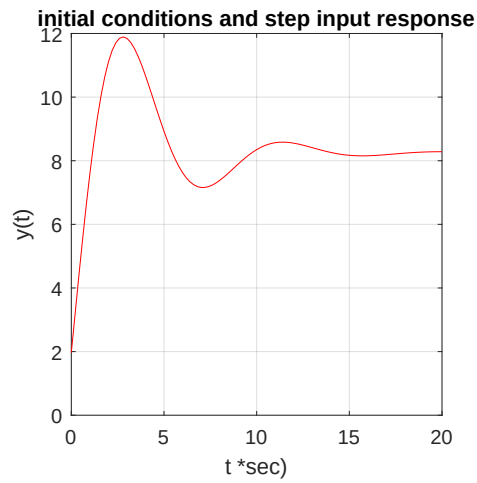
$$\left(\begin{array}{cc|c} -0.5572 & -0.7814 & 1 \\ 0.7814 & 0 & 0 \\ \hline 1.9691 & 6.4493 & 0 \end{array}\right) \mathcal{S}$$

```
x0 = {1,0};
y  = OutputResponse[{sys,x0},
         UnitStep[t],{t,0,20}];

Plot[y,{t,0,20},
  PlotRange->{{0,20},{0,13}},
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  Frame->True,
  ImageSize->300,
  AspectRatio->1,
  FrameLabel->{{"y(t)",None},
  {"t",
  "system response to initial\
      conditions and step input"}},
  RotateLabel->False,
  PlotStyle->Red]
```



Matlab

```
A = [-0.5572   -0.7814;0.7814  0];
B = [1;0];
C = [1.9691  6.4493];
x0 = [1 ; 0];

sys = ss(A,B,C,[]);

[y1,t] = initial(sys,x0,20);
y2    = step(sys,t);

plot(t,y1+y2,'r');

title(['initial conditions and step'...
          'input response']);
xlabel('t *sec)'); ylabel('y(t)');
grid
set(gcf,'Position',[10,10,320,320]);
```



1.31 Draw the root locus from the open loop transfer function

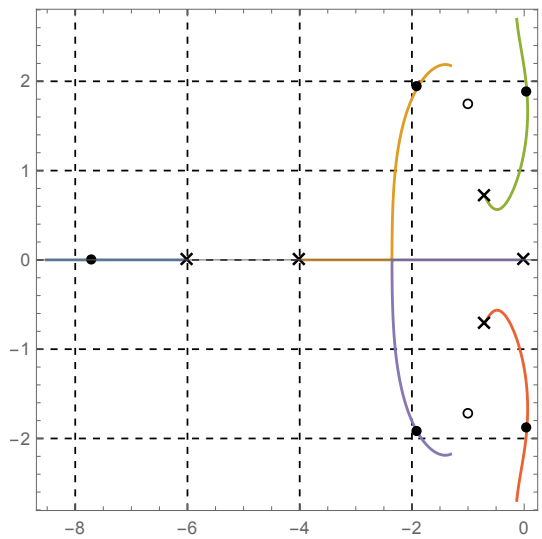
Problem: Given $L(s)$, the open loop transfer function, draw the root locus. Let

$$L(s) = \frac{s^2 + 2s + 4}{s(s+4)(s+6)(s^2 + 1.4s + 1)}$$

Root locus is the locus of the closed loop dominant pole as the gain k is varied from zero to infinity.

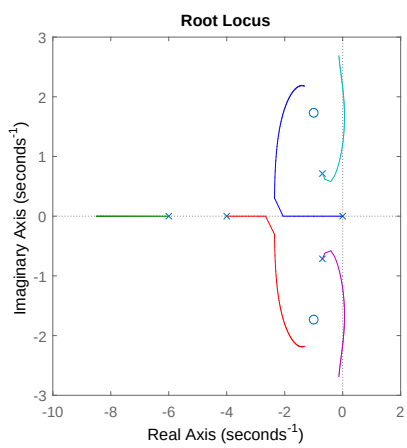
Mathematica

```
Remove["Global`*"]
sys = TransferFunctionModel[
 k*(s^2+2 s+4)/
     (s(s+4)(s+6)(s^2+1.4s+1)),s];
RootLocusPlot[sys,{k,0,100},
    ImageSize->300,
    GridLines->Automatic,
    GridLinesStyle->Dashed,
    Frame->True,
    AspectRatio -> 1]
```



Matlab

```
clear all; close all;
s=tf('s');
sys=(s^2+2*s+4)/...
 (s*(s+4)*(s+6)*(s^2+1.4*s+1));

rlocus(sys,0:100)
set(gcf,'Position',[10,10,420,420]);
```



1.32 Find e^{At} where A is a matrix

Mathematica

```
ClearAll["Global`*"]
mat={{0,1},
     {-2,-3}}
MatrixExp[mat t];
MatrixForm[%]
```

$$\begin{pmatrix} -e^{-2t}+2e^{-t} & -e^{-2t}+e^{-t} \\ 2e^{-2t}-2e^{-t} & 2e^{-2t}-e^{-t} \end{pmatrix}$$

Now verify the result by solving for e^{At} using the method would one would do by hand, if a computer was not around. There are a number of methods to do this by hand. The eigenvalue method, based on the Cayley Hamilton theorem will be used here. Find the eigenvalues of $|A-\lambda I|$

```
m = mat-lambda IdentityMatrix[Length[mat]]
```

$$\begin{pmatrix} -\lambda & 1 \\ -2 & -\lambda-3 \end{pmatrix}$$

```
Det[m]
```

$$\lambda^2+3\lambda+2$$

```
sol=Solve[%==0,lambda]
```

```
    Out[15]= {{lambda->-2},{lambda->-1}}
```

```
eig1=lambda/.sol[[1]]
eig2=lambda/.sol[[2]]
```

```
      Out[16]= -2
      Out[17]= -1
```

```
(*setup the equations to find b0,b1*)
eq1 = Exp[eig1 t]==b0+b1 eig1;
eq2 = Exp[eig2 t]==b0+b1 eig2;
sol = First@Solve[{eq1,eq2},{b0,b1}]
```

$$\left\{\mathrm{b0}\to e^{-2t}\left(2e^{t}-1\right),\mathrm{b1}\to e^{-2t}\left(e^{t}-1\right)\right\}$$

```
(*Now find e^At*)
b0=b0/.sol[[1]]
```

$$e^{-2t}\left(2e^{t}-1\right)$$

```
b1=b1/.sol[[2]]
```

$$e^{-2t}\left(e^{t}-1\right)$$

```
b0 IdentityMatrix[Length[mat]]+b1 mat;
Simplify[%]
MatrixForm[%]
```

$$\begin{pmatrix} e^{-2t}(-1+2e^{t}) & e^{-2t}(-1+e^{t}) \\ -2e^{-2t}(-1+e^{t}) & -e^{-2t}(-2+e^{t}) \end{pmatrix}$$

The answer is the same given by Mathematica's command `MatrixExp[]`

Matlab

```
syms t
A=[0 1;-2 -3];
expm(t*A)
```

```
ans =

[2/exp(t)-1/exp(2*t),1/exp(t)-1/exp(2*t)]
[2/exp(2*t)-2/exp(t),2/exp(2*t)-1/exp(t)]
```

```
pretty(ans)
```

```
+-                                          -+
| 2 exp(-t)-  exp(-2 t),exp(-t)-exp(-2 t)  |
|                                           |
| 2 exp(-2 t)-2 exp(-t),2 exp(-2 t)-exp(-t)|
+-                                          -+
```

Maple

```
restart;
A:=Matrix([[0,1],[-2,-3]]);
LinearAlgebra:-MatrixExponential(A,t);
```

$$\begin{bmatrix} -e^{-2t} + 2\,e^{-t} & e^{-t} - e^{-2t} \\ -2\,e^{-t} + 2\,e^{-2t} & 2\,e^{-2t} - e^{-t} \end{bmatrix}$$

1.33 Draw root locus for a discrete system

Problem: Given the open loop for a continuous time system as

$$sys = \frac{5s+1}{s^2+2s+3}$$

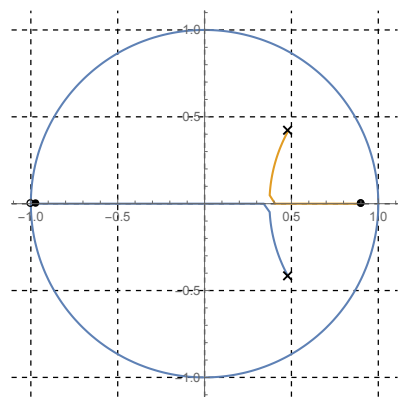
convert to discrete time using a sampling rate 0.5 second, and display the root locus for the discrete system.

Mathematica

```
Remove["Global`*"];
sys=TransferFunctionModel[
       k(5s+1)/(s^2+2s+3),s];

samplePeriod=0.5; (*sec*)
sysd=ToDiscreteTimeModel[sys,samplePeriod,z]
```

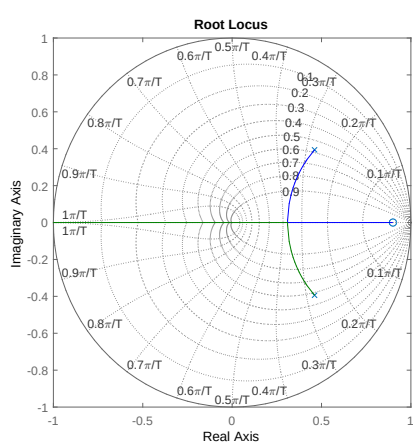
$$\left(\frac{-19.\,k+2.\,k\,z+21.\,k\,z^2}{11.-26.\,z+27.\,z^2}\right)_{\mathcal{T}\,0.5}$$

```
p1 = RootLocusPlot[sysd,{k,0,100},
       ImageSize->300,
       GridLines->Automatic,
       GridLinesStyle->Dashed,
       Frame->True,
       AspectRatio->1,
       PlotPoints->200,
       PlotStyle->PointSize[0.01]];

p2=ParametricPlot[{Sin[t],Cos[t]},{t,0,2Pi},
       AspectRatio->1,
       GridLines->Automatic,
       GridLinesStyle->Dashed];

Show[p2,p1]
```



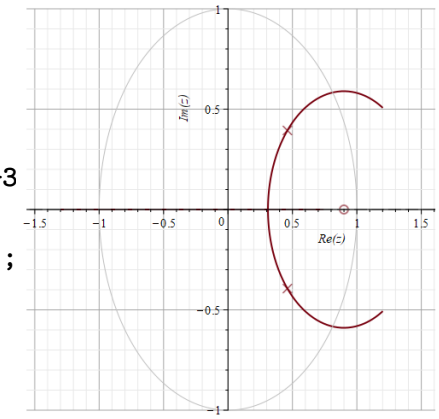
Matlab

```
clear all; close all;
s    = tf('s');
sys  = (5*s+1)/(s^2+2*s+3);
Ts   = 0.5; %sample time
sysd = c2d(sys,Ts,'zoh');

rlocus(sysd)
grid
```



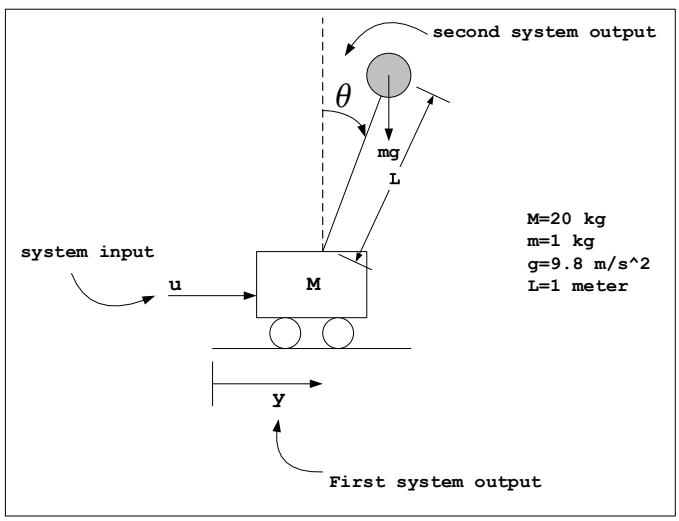
Maple

```
restart;
alias(DS=DynamicSystems):
sys := DS:-TransferFunction((5*s+1)/(s^2+2*s+3
T   :=1/2;  #will fail if 0.5 is used !!
sys := DS:-ToDiscrete(sys, T, 'method'='zoh');
DS:-RootLocusPlot(sys,-1..1,gridlines);
```



1.34 Plot the response of the inverted pendulum problem using state space

Problem: Given the inverted pendulum shown below, use state space using one input (the force on the cart) and 2 outputs (the cart horizontal displacement, and the pendulum angle. Plot each output separately for the same input.



Cart with inverted pendulum. One input, 2 outputs

Mathematica

```
Remove["Global`*"];
a = {{0, 1, 0, 0},
     {0, 0, ((-m)*g)/M, 0},
```

```
    {0, 0, 0, 1},
    {0, 0, ((M + m)*g)/(M*L), 0}};
```

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{gm}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g(m+M)}{LM} & 0 \end{pmatrix}$$

```
b = {{0}, {1/M}, {0}, {-(1/(M*L))}};
MatrixForm[b]
```

$$\begin{pmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{LM} \end{pmatrix}$$

```
c = {{1, 0, 0, 0}, {0, 0, 1, 0}};
MatrixForm[b]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

```
sys=StateSpaceModel[{a,b ,c}]
```

$$\left(\begin{array}{cccc|c} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{g\,m}{M} & 0 & \frac{1}{M} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{g\,(m+M)}{L\,M} & 0 & -\frac{1}{L\,M} \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right) \mathcal{S}$$

It is now possible to obtain the response of the system as analytical expression or an interpolatingFunction.

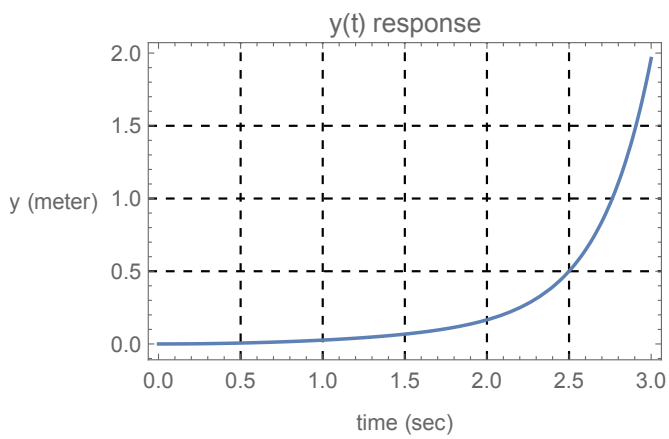
It is much more efficient to obtain the response as interpolatingFunction. This requires that the time domain be given.

Here is example of obtaining the analytical expression of the response

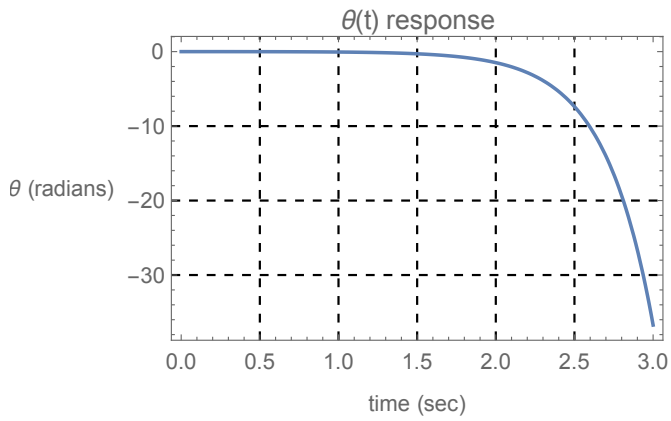
```
sys=sys/.{g->9.8,m->1,M->20,L->1};
y=Chop[OutputResponse[sys,UnitStep[t],t]]
```

$$e^{-6.41561t}(0.0238095e^{6.41561t}t^2\theta(t)+0.000115693e^{3.2078t}\theta(t)-0.000231385e^{6.41561t}\theta(t)+0.000115693e^{9.62341t}\theta($$
$$,e^{-6.41561t}(-0.00242954e^{3.2078t}\theta(t)+0.00485909e^{6.41561t}\theta(t)-0.00242954e^{9.62341t}\theta(t))$$

```
Plot[y[[1]],{t,0,3},
  PlotLabel->"y(t) response",
  FrameLabel->{"time (sec)","y (meter)"},
  Frame->True,
  PlotRange->All,
  ImageSize->300,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  RotateLabel->False]
```

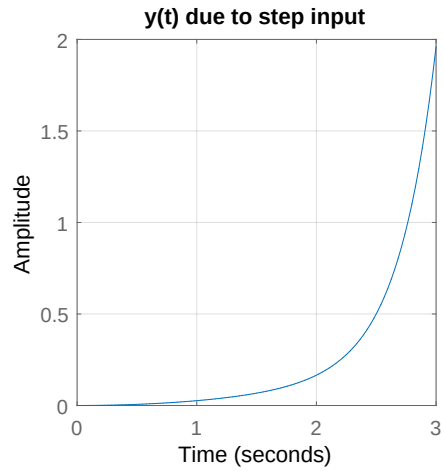



```
Plot[ y[[2]],{t,0,3},
 PlotLabel->"\[Theta](t) response",
 FrameLabel->{"time (sec)","\[Theta] (radians)"},
 Frame->True,
 PlotRange->All,
 ImageSize->300,
 GridLines->Automatic,
 GridLinesStyle->Dashed,
 RotateLabel->False]
```

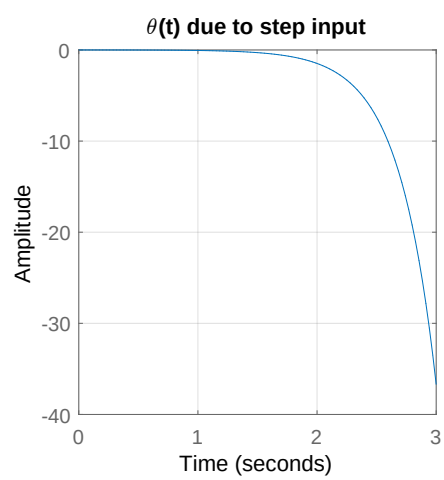


Matlab

```
clear all; close all;

m=1; M=20; L=1; g=9.8;

A=[0 1      0           0;
   0 0   -m*g/M         0;
   0 0      0           1;
   0 0  (M+m)*g/(M*L)   0
   ];
B=[0  1/M  0 -1/(M*L)]';
C=[1 0 0 0;
   0 0 1 0];
D=[0];

sys=ss(A,B,C(1,:),0);
step(sys,3);
grid on;
set(gcf,'Position',[10,10,320,320]);
title('y(t) due to step input');
```



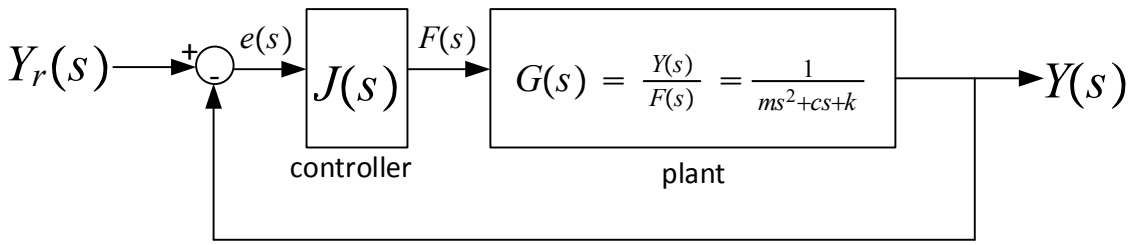

```
figure;
sys=ss(A,B,C(2,:),0);
step(sys,3);
title('\theta(t) due to step input');
grid on;
set(gcf,'Position',[10,10,320,320]);
```



1.35 How to build and connect a closed loop control systems and show the response?

1.35.1 example 1, single input, single output closed loop

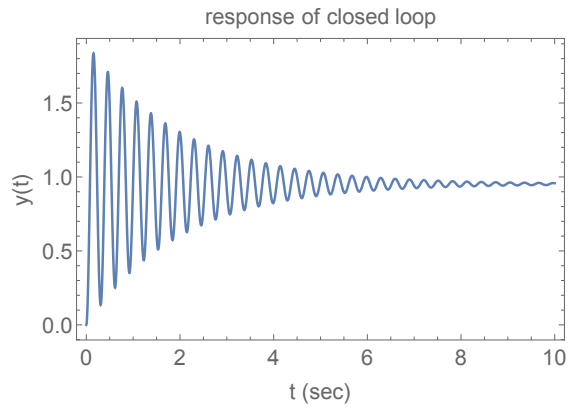
Given the following simple closed loop system, show the step response. Let mass $M = 1$kg, damping coefficient $c = 1$newton-seconds per meter and let the stiffness coefficient be $k = 20$newton per meter.



Using propertional controller $J(s) = k_p$ where $k_p = 400$. First connect the system and then show $y(t)$ for 5 seconds when the reference $y_r(t)$ is a unit step function.

Mathematica

```
m = 1; c = 1; k = 20; kp = 400;
plant      = TransferFunctionModel[1/(m s^2 + c s + k), s];
controller = TransferFunctionModel[kp, s];
sys        = SystemsModelSeriesConnect[plant, controller];
sys        = SystemsModelFeedbackConnect[sys];
o          = OutputResponse[sys, UnitStep[t], t];
Plot[o, {t, 0, 10}, Frame -> True, PlotRange -> All,
  FrameLabel -> {{"y(t)", None},
    {"t (sec)", "response of closed loop"}},
    BaseStyle -> 14]
```



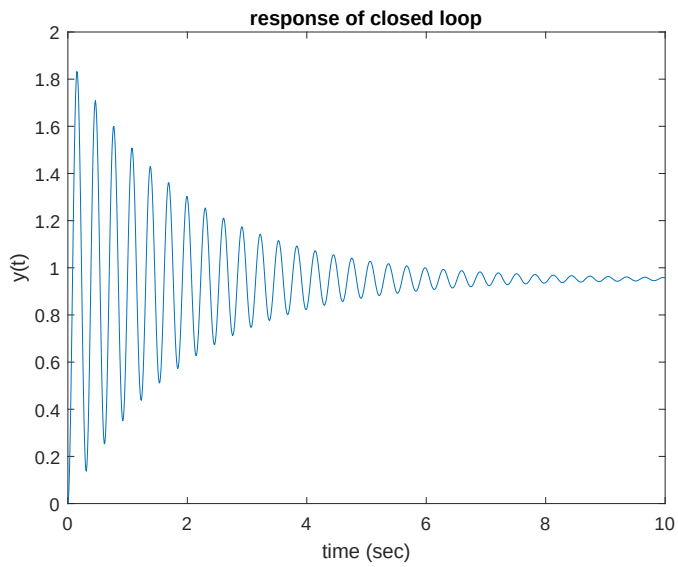
Matlab

```
close all;
m = 1; c = 1; k = 20;
s          = tf('s');
plant      = 1/(m*s^2+c*s+k);
controller = 400;
sys        = series(controller,plant);
sys        = feedback(sys,1);
[Y,T]      = step(sys,0:.01:10);
plot(T,Y);
xlabel('time (sec)');
ylabel('y(t)');
title('response of closed loop');
```

Another way to do the above is

```
m=1;
c=1;
k=20;
s=tf('s');
sys3=1/(m*s^2+c*s+k);
sys2=400;
sys1=1;
sys=append(sys1,sys2,sys3);
Q=[2 1 -3;3 2 0];
input=[1];
output=[2 3];
sys=append(sys1,sys2,sys3);
sys=connect(sys,Q,input,output);
```

```
T=0:.01:10;
[Y,T,X]=step(sys,T);
```



1.36 Compare the effect on the step response of a standard second order system as ζ changes

Problem: Given a standard second order system defined by the transfer function

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Change ζ and plot the step response for each to see the effect of changing ζ (the damping coefficient).

It is easy to solve this using the step command in Matlab, and similarly in Mathematica and Maple. But here it is solved directly from the differential equation.

The transfer function is written as

$$\frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Where $Y(s)$ and $U(s)$ are the Laplace transforms of the output and input respectively.

Hence the differential equation, assuming zero initial conditions becomes

$$y''(t) + 2\zeta\omega_n\, y'(t) + \omega_n^2\, y(t) = \omega_n^2\, u(t)$$

To solve the above in Matlab using ode45, the differential equation is converted to 2 first order ODE's as was done before resulting in

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} u(t)$$

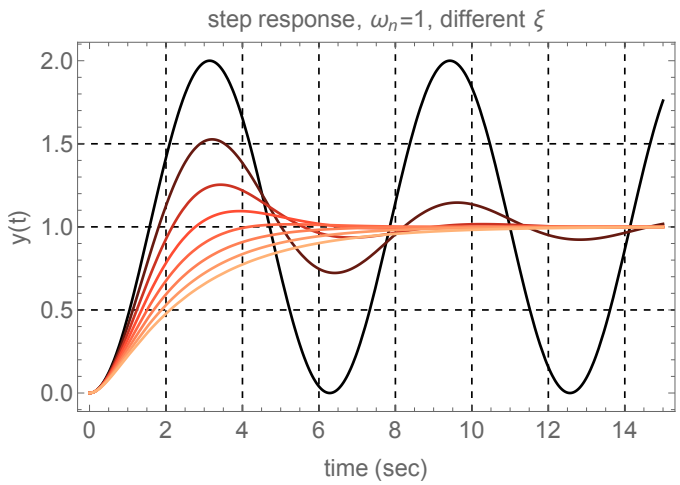
For a step input, $u(t) = 1$, we obtain

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix}$$

Now ODE45 can be used. In Mathematica the differential equation is solved directly using DSolve and no conversion is needed.

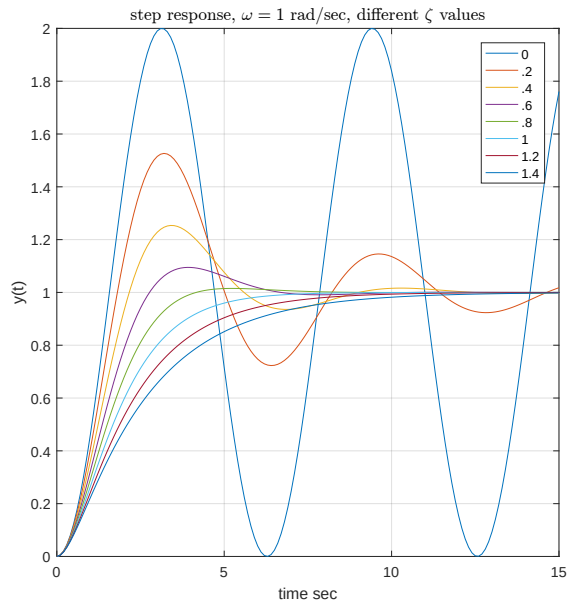
Mathematica

```
solve[z_]:=Module[{sol,eq,y,t,w=1},
  eq=y''[t]+2 z w y'[t]+
     w^2 y[t]==w^2 UnitStep[t];
  sol=First@NDSolve[{
   eq,y[0]==0,y'[0]==0},y[t],{t,0,15}
];

Plot[y[t]/.sol,{t,0,15},
   PlotPoints->30,
   PlotRange->All,
   Frame->True,
   FrameLabel->{{"y(t)",None},{"time (sec)",
   "step response,Subscript[\[Omega], n]=1"
      <>", different \[Xi]"}},
    ImageSize->400,
    PlotStyle->RGBColor[2z,z/2,z/3],
    LabelStyle->Directive[14]]
  ];

zeta = Range[0,1.4,.2];
r    = Map[solve,zeta];
Show[r,GridLines->Automatic,
     GridLinesStyle->Dashed]
```


step response, ω_n=1, different ξ
2.0
1.5
1.0
0.5
0.0
y(t)
0
2
4
6
8
10
12
14
time (sec)

Matlab

```
function e73
close all; clear all;

t     = 0:0.1:15;
ic    = [0 0]';
zeta  = 0:.2:1.4;
omega = 1;
sol   = zeros(length(t),length(zeta));

for i = 1:length(zeta)
    [t,y] = ode45(@f,t,ic,[],...
              zeta(i),omega);
    sol(:,i) = y(:,1);
end

plot(t,sol);
legend('0','.2','.4',...
 '.6','.8','1','1.2','1.4');
title('step response, $\omega=1$ rad/sec,'...
 'different $\zeta$ values',...
 'interpreter','latex',...
 'fontsize',12);

ylabel('y(t)');
xlabel('time sec');
grid on;
set(gcf,'Position',[10,10,600,600]);

function xdot=f(t,x,zeta,omega)
xdot=[x(2)
 -omega^2*x(1)-2*zeta*omega*x(2)]+...
 [0
 omega^2];
```



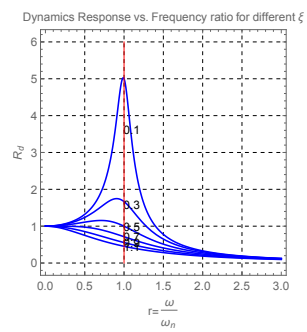
1.37 Plot the dynamic response factor R_d of a system as a function of $r = \frac{\omega}{\omega_n}$ for different damping ratios

Problem: Plot the standard curves showing how the dynamic response R_d changes as $r = \frac{\omega}{\omega_n}$ changes. Do this for different damping ratio ξ. Also plot the phase angle.

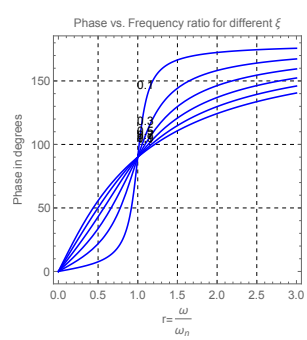
These plots are result of analysis of the response of a second order damped system to a harmonic loading. ω is the forcing frequency and ω_n is the natural frequency of the system.

Mathematica

```
Rd[r_,z_]:=1/Sqrt[(1-r^2)^2+(2 z r)^2];

phase[r_,z_]:=Module[{t},
   t=ArcTan[(2z r)/(1-r^2)];
   If[t<0,t=t+Pi];
   180/Pi t
];

plotOneZeta[z_,f_] := Module[{r,p1,p2},
 p1 = Plot[f[r,z],{r,0,3},PlotRange->All,
       PlotStyle->Blue];

 p2 = Graphics[Text[z,{1.1,1.1f[1.1,z]}]];
 Show[{p1,p2}]
];

p1 = Graphics[{Red,Line[{{1,0},{1,6}}]}];
p2 = Map[plotOneZeta[#,Rd]&,Range[.1,1.2,.2]];

Show[p2,p1,
 FrameLabel->{{"Subscript[R, d]",None},
 {"r= \[Omega]/Subscript[\[Omega], n]",
  "Dynamics Response vs. Frequency\
    ratio for different \[Xi]"}},
 Frame->True,
 GridLines->Automatic,GridLinesStyle->Dashed,
 ImageSize -> 300,AspectRatio -> 1]
```




```
p = Map[plotOneZeta[#,phase]&,Range[.1,1.2,.2]];
Show[p,FrameLabel->{{"Phase in degrees",None},
  {"r= \[Omega]/Subscript[\[Omega], n]",
  "Phase vs. Frequency ratio for different \[Xi]"}},
  Frame->True,
  GridLines->Automatic,GridLinesStyle->Dashed,
  ImageSize->300,AspectRatio->1]
```

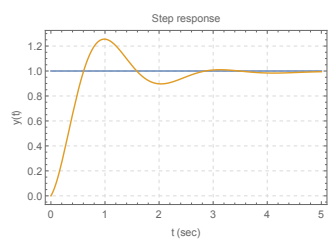


1.38 How to find closed loop step response to a plant with a PID controller?

Find and plot the step response of the plant $\frac{1}{s^2+2s+1}$ connected to a PID controller with $P = 10, I = 3.7, D = 0.7$. Use negative closed loopback.

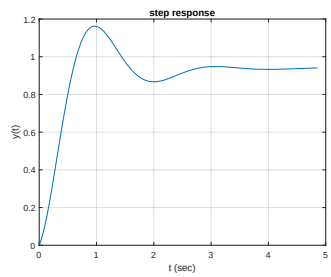
Mathematica

```
plant= TransferFunctionModel[1/(s^2 + 2*s + 1),s];
kip  = 10; ki = 3.7; kid = 0.7;
pid  = TransferFunctionModel[
          (kip*s + ki + kid*s^2)/s, s];
openLoop   = SystemsModelSeriesConnect[
   TransferFunctionModel[plant], pid];
closedLoop = SystemsModelFeedbackConnect[
                  openLoop];
input  = UnitStep[t];
output = OutputResponse[closedLoop, input, t];

Plot[{input, output}, {t, 0, 5}, PlotRange ->
 All, GridLines -> Automatic,
 GridLinesStyle -> Directive[LightGray, Dashed],
 Frame -> True,
 FrameLabel -> {{"y(t)", None},
    {"t (sec)", "Step response"}},
 BaseStyle -> 12]
```



Matlab

```
close all; clear;
s        = tf('s');
plant    = 1/(s^2 + 2*s + 1)
kip      = 10; ki = 3.7; kid = 0.7;
thePID   = pid(kip,kid,kid);
BLKSYS   = append(thePID,plant);
Q        = [2 1; 1 -2];
closedLoop = connect(BLKSYS,Q,1,2);
[y,t]   = step(closedLoop);
plot(t(1:140),y(1:140));
xlabel('t (sec)'); ylabel('y(t)');
title('step response');
grid
```



1.39 How to make Nyquist plot?

Nyquist command takes as input the open loop transfer function (not the closed loop!) and generates a plot, which was can look at to determine if the closed loop is stable or not. The closed loop is assumed to be unity feedback. For example, if the open loop is $G(s)$, then we know that the closed loop transfer function is $\frac{G(s)}{1+G(s)}$. But we call Nyquist with $G(s)$.

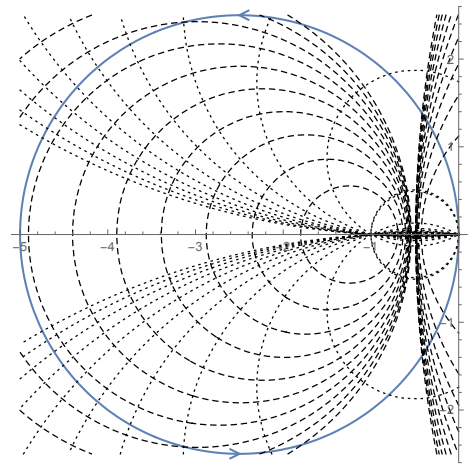
Here are two examples.

1.39.1 Example 1

Given $G(s) = \frac{s}{1-0.2s}$ generate Nyquist plot.

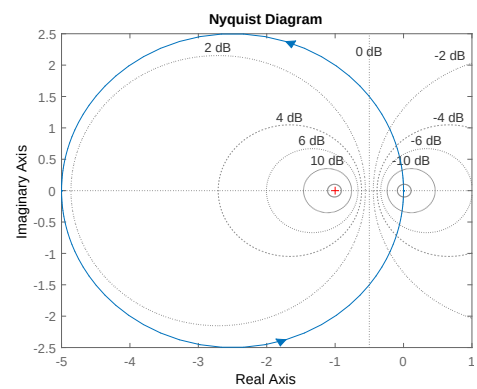
Mathematica

```
NyquistPlot[s/(1 - 0.2 s),
   NyquistGridLines -> Automatic]
```



Matlab

```
clear all; close all;
nyquist([1 0],[-0.2 1])
grid
```

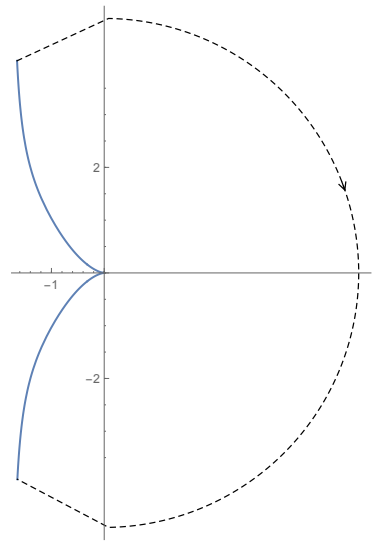


1.39.2 Example 2

Given $G(s) = \frac{5(1-0.5s)}{s(1+0.1s)(1-0.25s)}$ generate Nyquist plot.

Mathematica

```
NyquistPlot[5(1-0.5 s)/
    (s(1 + 0.1 s)(1 - 0.25 s))]
```

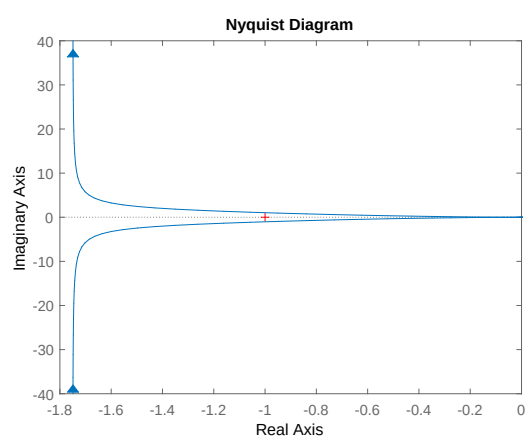


Matlab

```
clear all; close all;
s=tf('s');
sys=5*(1-0.5*s)/...
  (s*(1+0.1*s)*(1-0.25*s))
```

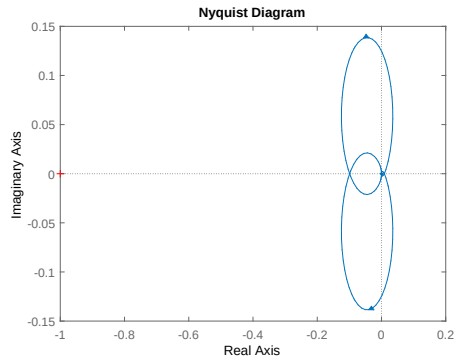
```
        2.5 s - 5
  ------------------------
  0.025 s^3 + 0.15 s^2 - s

Continuous-time transfer function.
```

```
nyquist(sys)
```



However, there is a better function to do this called nyquist1.m which I downloaded and tried. Here is its results

```
clear all; close all;
nyquist1()
```



CHAPTER 2

Linear algebra, Linear solvers, common operations on vectors and matrices

2.1 introduction

Mathematica and Matlab allow one to do pretty much the same operations in the area of linear algebra and matrix manipulation. Two things to keep in mind is that Mathematica uses a more general way to store data.

Mathematica uses ragged arrays or a list of lists. This means rows can have different sizes. (these are the lists inside the list). So a Mathematica matrix is stored in a list of lists. This is similar in a way to Matlab cell data structure, since each raw can have different length. In a standard `matrix` each row must have the same length.

In Matlab one can also have ragged arrays, these are called cells. In Mathematica, there is one data structure for both.

Another thing to keep in mind is that Matlab, due to its Fortran background is column major when it comes to operations on matrices. This simple example illustrate this difference. Suppose we have a matrix A of 3 rows, and want to find the location where $A(i,j) = 2$ where i is the row number and j is the column number. Given this matrix

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 5 \\ 5 & 6 & 7 & 2 \end{pmatrix}$$

Then the result of using the `find()` command in Matlab is

```
A=[1 2 3 4;
   2 3 1 5;
   5 6 7 2];
[I,J]=find(A==2)
```

```
I =
     2
     1
     3
J =
     1
     2
     4
     ]
```

The Matlab result gives the order of the rows it found the element at based on searching column wise since it lists the second row first in its result. Compare this to Mathematica `Position[]` command

```
mat = {{1, 2, 3, 4},
       {2, 3, 1, 5},
       {5, 6, 7, 2}}
Position[mat, 2]
```

which gives

```
{{1,2},
 {2,1},
 {3,4}}
```

Mathematica searched row-wise.

I found Mathematica for matrix operations takes more time to get familiar with compared to Matlab's, since Mathematica data structure is more general and therefore a little more complex (ragged arrays, or list of lists is its main data structure) compared to Matlab's. This is because Mathematica has to also support symbolics in its commands and not just numerical data.

In Maple the following short cuts can be used enter vectors and matrices: For row vector: `v:=<1|2|3|4>` and for column vector `v:=<1,2,3,4>` and for a Matrix of say 2 rows and 3 columns `A:=<1,2|3,4|5,6>`. Here `|` acts as a column separator. For transpose of matrix, `A^%T` is the short cut notation.

2.2 Multiply matrix with a vector

How to perform the following matrix/vector multiplication?

```
[ 1 2 3 ]   [ 1 ]
[ 4 5 6 ] * [ 2 ]
[ 7 8 8 ]   [ 3 ]
```

In Mathematica the dot "." is used for Matrix multiplication. In Matlab the "*" is used. In Fortran, MATMUL is used.

Mathematica

```
a={{1,2,3},{4,5,6},{7,8,8}};
x={1,2,3};

a.x
```

```
Out[7]= {14,32,47}
```

Matlab

```
A=[1 2 3;4 5 6;7 8 9];
x=[1; 2; 3];
c=A*x
```

```
    14
    32
    50
```

Julia

Same exact code as Matlab.

```
julia> A=[1 2 3;4 5 6;7 8 9]
3×3 Matrix{Int64}:
 1  2  3
 4  5  6
 7  8  9

julia> x=[1;2;3]
3-element Vector{Int64}:
 1
 2
 3

julia> c=a*x
3-element Vector{Int64}:
 14
 32
 50
```

Ada

```
with Ada.Text_Io; use Ada.Text_Io;
with Ada.Numerics.Real_Arrays;
     use Ada.Numerics.Real_Arrays;

procedure t1 is
  A : constant real_matrix :=
          (( 1.0,  2.0,  3.0),
          ( 4.0,  5.0,  6.0),
          ( 7.0,  8.0,  9.0));

  V : constant real_vector := (1.0,2.0,3.0);
  procedure Put (X : real_vector) is
    begin
        FOR e of X LOOP
            put_line(float'image(e));
        END LOOP;
  end put;
begin
    put(A*v);
end t1;
```

compile and run

```
>gnatmake -gnat2012 t1.adb
gcc -c -gnat2012 t1.adb
gnatbind -x t1.ali
gnatlink t1.ali
>./t1
 1.40000E+01
 3.20000E+01
 5.00000E+01
```

Fortran

```
 PROGRAM main
IMPLICIT NONE
integer, parameter :: dp = kind(1.D0)
integer, parameter :: N=3
real(kind=dp), parameter, DIMENSION(N, N) :: &
  A=DBLE(reshape([ 1.0, 2.0, 3.0,&
                   4.0, 5.0, 6.0,&
                   7.0, 8.0, 9.0], shape(A), &
                                   order=[2,1]))

real(kind=dp), parameter, DIMENSION(N, 1) :: &
           x=DBLE(reshape([1.0, 2.0, 3.0], shape(x)))

real(kind=dp), DIMENSION(N, 1) :: result
integer, DIMENSION(2)    :: siz

result = MATMUL(A,x)
siz    = SHAPE(result)
Write(*,*) result

print *, "number of rows = ",siz(1)
print *, "number of columns = ",siz(2)

end program
```

compile and run

```
>gfortran  -fcheck=all -Wall -Wconversion
   -Wextra -Wconversion-extra -pedantic test.f90
>./a.out
14.000000000000000  32.000000000000000 50.000000000000000
 number of rows =            3
 number of columns =            1
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]):
x:=Vector([1,2,3]): #default is column vector
c:=A.x;
```

$$\begin{bmatrix} 14 \\ 32 \\ 50 \end{bmatrix}$$

Python

```
import numpy as np
mat=np.array([[1,2,3],[4,5,6],[7,8,8]])
b=np.array([1,2,3])
b.shape=(3,1)
r=dot(mat,b)
```

```
array([[14],
       [32],
       [47]])
```

2.3 Insert a number at specific position in a vector or list

The problem is to insert a number into a vector given the index.

Mathematica

```
vec={1,2,3,4};
vec=Insert[vec,99,3];
vec
```

```
Out[11]= {1,2,99,3,4}
```

Matlab

```
A=[1 2 3 4];
A=[ A(1:2) 99 A(3:end) ]
```

```
A =
     1     2    99     3     4
```

Julia

```
A=[1,2,3,4];
A=insert!(A,3,99)'
```

```
1Œ5 adjoint(::Vector{Int64}) with eltype Int64:
 1  2  99  3  4
```

Fortran

```
 program f
  implicit none
  REAL(4), ALLOCATABLE :: A(:)
  A=[1,2,3,4];
  A=[A(1:2), 99.0, A(3:)]
  Print *,A
end program f
```

```
>gfortran -std=f2008 t2.f90
>./a.out
 1.0000000 2.0000000 99.000000
          3.0000000 4.0000000
```

Maple

```
v:=Vector[row]([1,2,3,4]);
v:=Vector[row]([v[1..2],99,v[3..]]);
```

Or Using <> notation

```
v:=<1|2|3|4>;
v:=<v(1..2)|99|v(3..)>;
```

```
v :=[ 1 2 99 3 4]
```

Python

Python uses zero index.

```
import numpy as np
b=np.array([1,2,3,4])
b=numpy.insert(b,2,99)
b
```

```
Out[86]: array([ 1,  2, 99,  3,  4])
```

2.4 Insert a row into a matrix

The problem is to insert a row into the second row position in a 2D matrix

Mathematica

```
mat={{1,2,3},
      {4,5,6},
      {7,8,9}}
mat = Insert[mat,{90,91,92},2]
```

```
{{1,2,3},
 {90,91,92},
 {4,5,6},
 {7,8,9}}
```

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9]
A=[ A(1,:); [90 91 92];  A(2:end,:) ]
```

```
     1     2     3
    90    91    92
     4     5     6
     7     8     9
```

Maple

Using <<>> notation

```
A:=< <1|2|3>,
     <4|5|6>,
     <7|8|9>>;

A:=< A(1..2,..),<90|91|92>, A(3,..)>;
```

```
[ 1  2   3
  4  5   6
  90 91  92
  7  8   9]
```

Using `Matrix/Vector`

```
A:=Matrix([ [1,2,3],[4,5,6],[7,8,9]]);
A:=Matrix([ [A[1..2,..]],
            [Vector[row]([91,92,92])],
            [A[3,..]
          ] ]);
```

Python

```
import numpy as np
mat=np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])
mat=numpy.insert(mat,1,[90,91,92],0)
```

```
array([[ 1,  2,  3],
       [90, 91, 92],
       [ 4,  5,  6],
       [ 7,  8,  9]])
```

Fortran

```
 program f
  implicit none
  integer i
  integer, allocatable  :: A(:,:)
  integer, allocatable  :: B(:)
  B =[90,91,92]
  A = reshape ([1,2,3, &
                4,5,6, &
                7,8,9], [3,3], order=[2,1])


  A=reshape([A(1,:),  &
            B, &
            A(2:,:)], [4,3],order=[2,1])

  do i=LBOUND(A, 1),UBOUND(A, 1)
     print *,A(i,:)
  end do
end program f
```

Compile and run

```
>gfortran f.f90
>./a.out
 1           2           3
 90          91          92
 4           7           5
 8           6           9
```

2.5 Insert a column into a matrix

The problem is to insert a column into the second column position in a 2D matrix.

Mathematica

```
mat = {{1, 2, 3},
       {4, 5, 6},
       {7, 8, 9}};

mat = Insert[Transpose[mat],
         {90, 91, 92}, 2];
mat = Transpose[mat]
```

```
{{1,90,2,3},
 {4,91,5,6},
 {7,92,8,9}}
```

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9];

A=[A(:,1)  [90 91 92]'   A(:,2:end) ]
```

```
     1    90     2     3
     4    91     5     6
     7    92     8     9
```

Fortran

```
program t4
 implicit none
 integer(4) ::i
 integer(4), ALLOCATABLE ::A(:,:)

 A = reshape ([1,2,3,   &
               4,5,6,   &
               7,8,9], [3,3], order=[2,1])

 A = reshape ([A(:,1), [90,91,92] ,
                 A(:,2:)] , [3,4])

 do i=LBOUND(A, 1),UBOUND(A, 1)
    print *,A(i,:)
 end do

end program t4
```

```
>gfortran t4.f90
>./a.out
 1          90          2          3
 4          91          5          6
 7          92          8          9
```

Maple

```
A:=< <1|2|3>,<4|5|6>,<7|8|9>>;
A:=< A(..,1)|<90,91,92>|A(..,2..)>;
```

Using `Matrix/Vector`

```
A:=Matrix([ [1,2,3],[4,5,6],[7,8,9]]);
A:=Matrix([ [A[..,1],
            Vector[column]([91,92,92]),
            A[..,2..] ]]);
```

```
[ 1    90    2    3
  4    91    5    6
  7    92    8    9]
```

Python

```
import numpy as np
mat=np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])

mat=numpy.insert(mat,1,[90,91,92],1)
```

```
array([[ 1, 90,  2,  3],
       [ 4, 91,  5,  6],
       [ 7, 92,  8,  9]])
```

2.6 Build matrix from other matrices and vectors

2.6.1 First example

Given column vectors $v1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $v2 = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$ and $v3 = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}$ and $v4 = \begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix}$

generate the matrix $m = \begin{bmatrix} v_1 & v_2 \\ v_3 & v_4 \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}$

Matlab was the easiest of all to do these operations with. No surprise, as Matlab was designed for Matrix and vector operations. But I was surprised that Maple actually had good support for these things, using its `<>` notation, which makes working with matrices and vectors much easier.

The command `ArrayFlatten` is essential for this in Mathematica.

Notice the need to use `Transpose[{v}]` in order to convert it to a column matrix. This is needed since in Mathematica, a list can be a row or column, depending on context.

Mathematica

```
v1 = {1, 2, 3}; v2 = {4, 5, 6};
v3 = {7, 8, 9}; v4 = {10, 11, 12};
m = ArrayFlatten[
 {{Transpose@{v1}, Transpose@{v2}},
 {Transpose@{v3}, Transpose@{v4}}}
     ]
```

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{pmatrix}$$

Matlab

```
v1=[1,2,3]'; v2=[4,5,6]';
v3=[7,8,9]'; v4=[10,11,12]';
m=[v1 v2;v3 v4]
```

```
m =
     1     4
     2     5
     3     6
     7    10
     8    11
     9    12
```

Maple

```
restart;
v1:=<1,2,3>; #column by default
v2:=<4,5,6>;
v3:=<7,8,9>;
v4:=<10,11,12>;
m:=< <v1|v2>,
     <v3|v4>>;

#or using Vector/Matrix notations
restart;
v1:=Vector([1,2,3]); #column by default
v2:=Vector([4,5,6]); #column by default
v3:=Vector([7,8,9]); #column by default
v4:=Vector([10,11,12]); #column by default
m:=Matrix([ [v1,v2],[v3,v4]])
```

$$\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix}$$

Python

```
import numpy as np
v1=np.array((1,2,3));
v2=np.array((4,5,6));
v3=np.array((7,8,9));
v4=np.array((10,11,12));

r1 =np.hstack([(v1,v2)]).T
r2 =np.hstack([(v3,v4)]).T
mat = np.vstack((r1,r2))
```

```
Out[211]:
array([[ 1,  4],
       [ 2,  5],
       [ 3,  6],
       [ 7, 10],
       [ 8, 11],
       [ 9, 12]])
```

Another way

```
v1=np.array([(1,2,3)]).T
v2=np.array([(4,5,6)]).T
v3=np.array([(7,8,9)]).T
v4=np.array([(10,11,12)]).T

mat =np.hstack((
     np.vstack((v1,v3)),
     np.vstack((v2,v4)))
    )
```

Fortran

```
PROGRAM main
  IMPLICIT none
  INTEGER :: i
  INTEGER , DIMENSION(3) :: v1,v2,v3,v4
  INTEGER , DIMENSION(6,2) :: m
  v1 = [1,2,3]; v2=[4,5,6];
  v3=[7,8,9];   v4=[10,11,12];

  m(1:3,1) = v1; m(1:3,2)=v2;
  m(4:,1)=v3;  m(4:,2)=v4;

  DO i=1,size(m,1)
     PRINT *, m(i,:)
  END DO

END PROGRAM main
```

```
>gfortran -Wall foo.f90
>./a.out
1           4
2           5
3           6
7          10
8          11
9          12
```

Using the RESHAPE command

```
PROGRAM main
  IMPLICIT none
  INTEGER :: i
  INTEGER , DIMENSION(3) :: v1,v2,v3,v4
  INTEGER , DIMENSION(6,2) :: m
  v1 = [1,2,3]; v2=[4,5,6];
  v3=[7,8,9]; v4=[10,11,12];

  m = RESHAPE([v1,v3,v2,v4], SHAPE(m), ORDER=[1,2])

  DO i=1,size(m,1)
     PRINT *, m(i,:)
  END DO

END PROGRAM main
```

2.6.2 second example

Given mix of matrices and vectors, such as $v1 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $v2 = \begin{bmatrix} 4 & 5 \\ 6 & 7 \\ 8 & 9 \end{bmatrix}$ and $v3 = \begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix}$ and $v4 = \begin{bmatrix} 13 \\ 14 \\ 15 \end{bmatrix}$ and $v5 = \begin{bmatrix} 16 \\ 17 \\ 18 \end{bmatrix}$

generate the matrix 6 by 3 matrix $m = \begin{bmatrix} v1 & v2 & \\ v3 & v4 & v5 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 5 \\ 2 & 6 & 7 \\ 3 & 8 & 9 \\ 10 & 13 & 16 \\ 11 & 14 & 17 \\ 12 & 15 & 18 \end{bmatrix}$

Mathematica, thanks for function by Kuba at Mathematica stackexachage, this becomes easy to do

Mathematica

```
myArrayFlatten = Flatten /@ Flatten[#, {{1, 3}}] &

v1 = {1, 2, 3};
v2 = {{4, 5}, {6, 7}, {8, 9}};
v3 = {10, 11, 12};
v4 = {13, 14, 15};
v5 = {16, 17, 18};

m={
  {v1, v2},
  {v3, v4, v5},
} // myArrayFlatten
```

Maple

```
restart;
```

```
v1:=<1,2,3>; #column by default
v2:=<<4|5>,
     <6|7>,
     <8|9>>;
v3:=<10,11,12>;
v4:=<13,14,15>;
v5:=<16,17,18>;
m:=< <v1|v2>,
     <v3|v4|v5>>;
```

Matlab

```
v1=[1,2,3]';
v2=[4,5;6,7;8,9];
v3=[10,11,12]';
v4=[13,14,15]';
v5=[16,17,18]';

m=[v1 v2;v3 v4 v5]
```

Fortran

```
 PROGRAM main
   IMPLICIT none
   INTEGER :: i
   INTEGER , DIMENSION(3)   :: v1,v3,v4,v5
   INTEGER , DIMENSION(3,2) :: v2 = RESHAPE([4,6,8,5,7,9],SHAPE(v2),ORDER=[1,2])
   INTEGER , DIMENSION(6,3) :: m
   v1 = [1,2,3];   v3=[10,11,12];
   v4 =[13,14,15]; v5=[16,17,18];
   m = RESHAPE([v1,v3,v2(:,1),v4,v2(:,2),v5], SHAPE(m), ORDER=[1,2])

   DO i=1,size(m,1)
      PRINT *, m(i,:)
   END DO

END PROGRAM main
```

```
>gfortran -Wall foo.f90
>./a.out
           1           4           5
```

```
          2          6          7
          3          8          9
         10         13         16
         11         14         17
         12         15         18
```

2.7 Generate a random 2D matrix from uniform (0 to 1) and from normal distributions

Mathematica

```
(*from uniform over 0,1*)
mat = RandomReal[
       UniformDistribution[{0,1}],{3,4}]
```

```
{{0.100843,0.356115,0.700317,0.657852},
 {0.238019,0.59598,0.523526,0.576362},
 {0.339828,0.32922,0.0632487,0.815892}}
```

```
(*from normal, zero mean, 1 std*)

mat=RandomReal[
   NormalDistribution[0,1],{3,4}]
```

```
{{-0.226424,1.19553,0.601433,1.28032},
{1.66596,0.176225,-0.619644,0.371884},
{0.895414,-0.394081,0.153507,-1.74643}}
```

Matlab

```
mat=rand(3,4)
```

```
mat =
 0.6787    0.3922    0.7060    0.0462
 0.7577    0.6555    0.0318    0.0971
 0.7431    0.1712    0.2769    0.8235
```

```
mat=randn(3,4)
```

```
mat =
 0.3252   -1.7115    0.3192   -0.0301
-0.7549   -0.1022    0.3129   -0.1649
 1.3703   -0.2414   -0.8649    0.6277
```

Maple

```
A:=ArrayTools:-RandomArray(3,4,
              distribution = uniform);

#just to round to 2 decimal points
parse~(sprintf~("%0.2f",A));
```

```
[[0.93,0.66,0.92,0.80],
 [0.85,0.96,0.42,0.49],
 [0.04,0.79,0.14,0.96]
]
```

Or

```
interface(displayprecision=5):
A:=ArrayTools:-RandomArray(3,4,
             distribution = uniform);
```

$$\begin{bmatrix} 0.970592781760615697 & 0.957506835434297598 & 0.0975404049994095246 & 0.126986816293506055 \\ 0.157613081677548283 & 0.546881519204983846 & 0.632359246225409510 & 0.905791937075619225 \\ 0.964888535199276531 & 0.278498218867048397 & 0.913375856139019393 & 0.814723686393178936 \end{bmatrix}$$

Fortran

```
 program t5
  implicit none
  INTEGER(4) ::i=0,j
  REAL ::A(3,3),mean,sd,pi,temp

  CALL RANDOM_SEED(i)
  CALL RANDOM_NUMBER(A)

  print *, 'from uniform distribution'
  do i=LBOUND(A, 1),UBOUND(A, 1)
     print *,A(i,:)
  end do

  !this block below uses the logic as explained in
  !http://rosettacode.org/wiki/Random_numbers#Fortran
  !
  mean=0.0
  sd=1.0
  pi = 4.0*ATAN(1.0)

  ! Now convert to normal distribution
    DO i = 1, SIZE(A,1)-1, 2
      DO j = 1, SIZE(A,2)-1, 2
         temp = sd * SQRT(-2.0*LOG(A(i,j))) * COS(2*pi*A(i+1,j+1)) + mean
         A(i+1,j+1) = sd* SQRT(-2.0*LOG(A(i,j)))*SIN(2*pi*A(i+1,j+1))+mean
         A(i,j) = temp
      END DO
   END DO

 print *, 'from normal distribution'
 do i=LBOUND(A, 1),UBOUND(A, 1)
      print *,A(i,:)
 end do

end program t5
```

```
>gfortran -Wall -std=f2008 t5.f90
>./a.out
```

```
from uniform distribution
 0.99755955      0.74792767      7.37542510E-02
 0.56682467      0.36739087      5.35517931E-03
 0.96591532      0.48063689      0.34708124

from normal distribution
-4.70122509E-02  0.74792767      7.37542510E-02
 0.56682467      5.17370142E-02  5.35517931E-03
 0.96591532      0.48063689      0.34708124
```

Did not find a build-in support for random numbers from normal distribution, need to look more.

2.8 Generate an n by m zero matrix

Mathematica

```
mat=Table[0,{3},{4}]
```

```
{{0,0,0,0},
 {0,0,0,0},
 {0,0,0,0}}
```

Matlab

```
A=zeros(3,4)
```

```
A =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Maple

```
LinearAlgebra:-ZeroMatrix(3,4);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Fortran

```
 program f
  implicit none
  integer i
  integer, parameter :: dp = kind(1.D0)
  real(kind=dp), DIMENSION(3, 4):: A

  A=0

  do i=LBOUND(A, 1),UBOUND(A, 1)
       print *,A(i,:)
  end do
end program f
```

```
tmp>gfortran -std=f2008 f.f90
tmp>./a.out
0.0000000000000000        0.0000000000000000        0.0000000000000000        0.00000000000
0.0000000000000000        0.0000000000000000        0.0000000000000000        0.00000000000
0.0000000000000000        0.0000000000000000        0.0000000000000000        0.00000000000
```

Ada

```
with Ada.Text_IO;              use Ada.Text_IO;
with Ada.Numerics.Real_Arrays; use Ada.Numerics.Real_Arrays;

procedure t1 is
   A : Real_Matrix (1 .. 3, 1 .. 4) := (others => (others => 0.0));
begin
   null;
end t1;
```

2.9 Rotate a matrix by 90 degrees

Mathematica

```
mat = {{1, 2, 3},
       {4, 5, 6},
       {7, 8, 9}};
mat = Reverse[Transpose[mat]]
```

```
    {{3,6,9},
     {2,5,8},
     {1,4,7}}
```

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9]
A=rot90(A)
```

```
     3     6     9
     2     5     8
     1     4     7
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
A:=ArrayTools:-FlipDimension(A,2);
```

```
[[3,2,1],
 [6,5,4],
 [9,8,7]]
```

```
A:=A^%T;
```

```
   [[3,6,9],
    [2,5,8],
    [1,4,7]]
```

Fortran

Using additional copy for the matrix

```
 program f
  implicit none
  integer(4) ::i,j
  integer(4), ALLOCATABLE ::A(:,:),B(:,:)

  A = reshape ([1,2,3,   &
                4,5,6,   &
                7,8,9],  &
               [3,3], &
               order=[2,1] &
              )
  B=A
  i=0

  do j=UBOUND(A, 2),LBOUND(A, 2),-1
     i=i+1
     B(i,:)=A(:,j)
  end do

  do i=LBOUND(B, 1),UBOUND(B, 1)
       print *,B(i,:)
  end do

end program f
```

```
mp>gfortran -std=f2008 f.f90
tmp>./a.out
           3           6           9
           2           5           8
           1           4           7
```

2.10 Generate a diagonal matrix with given values on the diagonal

Problem: generate diagonal matrix with $2, 4, 6, 8$ on the diagonal.

Mathematica

```
DiagonalMatrix[2 Range[1, 4]]
```

```
{{2,0,0,0},
 {0,4,0,0},
 {0,0,6,0},
 {0,0,0,8}}
```

Matlab

```
diag(2*(1:4))
```

```
2     0     0     0
0     4     0     0
0     0     6     0
0     0     0     8
```

Maple

```
A:=LinearAlgebra:-DiagonalMatrix([2,4,6,8]);
#or
A:=LinearAlgebra:-DiagonalMatrix(
                        [seq(i,i=2..8,2)]);
```

```
[[2,0,0,0],
 [0,4,0,0],
 [0,0,6,0],
 [0,0,0,8]]
```

2.11 Sum elements in a matrix along the diagonal

Mathematica

```
mat = {{1, 2, 3},
       {4, 5, 6},
       {7, 8, 9}}
Tr[mat]
```

```
Out[45]= 15
```

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9]
sum(diag(A))
```

```
ans =
    15
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
d:=MTM:-diag(A);
add(x,x in d);
```

Another ways

15

```
LinearAlgebra:-Diagonal(A);
Student:-LinearAlgebra:-Diagonal(A);
```

2.12 Find the product of elements in a matrix along the diagonal

Mathematica

```
mat= {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9}}
Apply[Times,Diagonal[mat]]
```

Out[49]= 45

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9]
prod(diag(A))
```

ans = 45

Maple

```
restart;
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
d:=MTM:-diag(A);
mul(x,x in d);

#or simpler is
restart;
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
mul(LinearAlgebra:-Diagonal(A));
```

45

2.13 Check if a Matrix is diagonal

A diagonal matrix is one which has only zero elements off the diagonal. The Mathematica code was contributed by Jon McLoone.

Mathematica

```
diagonalQ[m_List]/;ArrayDepth[m]===2&&Equal@@Dimensions[m]:=
  And@@Flatten[MapIndexed[#1===0||Equal@@#2&,m,{2}]];

diagonalQ[m_]:=Return[False];
matA = {{1, 2},
        {2, 4}};

matB = {{1, 0},
        {0, 2}};

matC = {{1, 0, 2},
        {0, 2, 4}};

diagonalQ[matA]
diagonalQ[matB]
diagonalQ[matC]
```

```
Out[59]= False
Out[60]= True
Out[61]= False
```

Maple

```
A:=Matrix([[1,0],[0,2]]);
Student:-NumericalAnalysis:-IsMatrixShape(A,'diagonal');
```

true

2.14 Find all positions of elements in a Matrix that are larger than some value

The problem is to find locations or positions of all elements in a matrix that are larger or equal than some numerical value such as 2 in this example.

Mathematica

```
mat = {{1, 2,3},
       {2,1,-5},
       {6,0,0}};

p = Position[mat, _?(#1 >= 2&)]
```

```
{{1,2},{1,3},{2,1},{3,1}}
```

Matlab

```
A=[1 2 3;
   2 1 -5;
   6 0 0]
[I,J]=find(A>=2)
```

```
I =
     2
     3
     1
     1
J =
     1
     1
     2
     3
```

Maple

Maple is little weak in this area. There does not seem to be a buildin function to return indices in matrix based on some condition. I searched ListTools and ArrayTools. So had to program it in.

```
A:=Matrix([[1,2,3],[2,1,-5],[6,0,0]]);
seq(seq(`if`(A[i,j]>=2,[i,j],NULL),i=1..3),j=1..3);
```

```
[2, 1], [3, 1], [1, 2], [1, 3]
```

2.15 Replicate a matrix

Given Matrix

```
    1     2
    3     4
```

Generate a new matrix of size 2 by 3 where each element of the new matrix is the above matrix. Hence the new matrix will look like

```
    1     2     1     2     1     2
    3     4     3     4     3     4
    1     2     1     2     1     2
    3     4     3     4     3     4
```

In Matlab, repmat() is used. In Mathematica, a Table command is used, followed by ArrayFlatten[]

Mathematica

```
mat = {{1,2},
       {3,4}}

r = Table[mat,{2},{3}];
ArrayFlatten[r]
```

```
{{1,2,1,2,1,2},
 {3,4,3,4,3,4},
 {1,2,1,2,1,2},
 {3,4,3,4,3,4}}
```

Matlab

```
A=[1 2;3 4]
repmat(A,2,3)
```

```
ans =
     1     2     1     2     1     2
     3     4     3     4     3     4
     1     2     1     2     1     2
     3     4     3     4     3     4
```

Another way is to use kron() in matlab, and KroneckerProduct in Mathematica and LinearAlgebra[KroneckerProduct] in Maple, which I think is a better way. As follows

Mathematica

```
mat = {{1,2},
       {3,4}}

kernel = Table[1, {2}, {3}];
   (* {{1, 1, 1},
      {1, 1, 1}} *)

KroneckerProduct[kernel, mat]
```

```
{{1,2,1,2,1,2},
 {3,4,3,4,3,4},
 {1,2,1,2,1,2},
 {3,4,3,4,3,4}}
```

Matlab

```
A=[1 2;3 4]
kernel=ones(2,3)

kernel =
     1     1     1
     1     1     1

kron(kernel,A)
```

```
ans =
     1     2     1     2     1     2
     3     4     3     4     3     4
     1     2     1     2     1     2
     3     4     3     4     3     4
```

Maple

```
A:=Matrix([[1,2],[3,4]]);
kern:=Matrix([[1,1,1],[1,1,1]]);
LinearAlgebra:-KroneckerProduct(kern,A);
```

```
[[1,2,1,2,1,2],
 [3,4,3,4,3,4],
 [1,2,1,2,1,2],
 [3,4,3,4,3,4]]
```

2.16 Find the location of the maximum value in a matrix

Mathematica

```
mat ={{2,4,-3},
      {7,9,5},
      {0,5.4,9}}

m = Max[mat];  (*this gives values *)
Position[mat,m] (*this find locations*)
```

```
Out[142]= {{2,2},{3,3}}
```

Matlab

```
h = [2, 4, -3;
     7, 9, 5;
     0, 5.4, 9]

m = max(h(:));

[r,c]=find(h==m)
```

```
r =
     2
     3
c =
     2
     3
```

Maple

This below finds position of first max.

```
A:=Matrix([[2,4,-3],[7,9,5],[0,5.4,9]]);
v:=max(A);
member(v,A,'pos');
pos
```

```
2,2
```

Maple support for such operations seems to be not as strong as Matlab. One way to find locations of all elements is by using explicit loop

```
A:=Matrix([[2,4,-3],[7,9,5],[0,5.4,9]]);
v:=max(A);
r:=[seq(seq(`if`(A[i,j]=v,[i,j],NULL),i=1..3),j=1..3)];
```

```
[[2, 2], [3, 3]]
```

2.17 Swap 2 columns in a matrix

Give a matrix

```
    1     2
    3     4
    5     6
```

How to change it so that the second column becomes the first, and the first becomes the second? so that the result become

```
    2     1
    4     3
    6     5
```

Mathematica

```
a={{1,2},
   {3,4},
   {5,6}}
Reverse[a,2]
```

```
Out[29]= {{2, 1},
          {4, 3},
          {6, 5}}
```

Matlab

```
a =[1 2;
    3 4;
    5 6]
[a(:,2) a(:,1)]
```

```
     2     1
     4     3
     6     5
```

Maple

```
A:=Matrix([[1,2],[3,4],[5,6]]);
A:=Matrix([ A[..,2], A[..,1] ])
```

```
[[2,1],
 [4,3],
 [6,5]]
```

2.18 Join 2 matrices side-by-side and on top of each others

Mathematica

In Mathematica, to join 2 matrices side-by-side, use Join with `'2'` as the third argument. To join them one on top of the other, use `'1'` as the third argument

```
a = RandomInteger[10, {3, 3}]
```

```
Out[146]= {{8,5, 1},
           {8,10,8},
           {6,0, 8}}
```

```
b = RandomInteger[10, {3, 3}]
```

```
{{9, 0,1},
{10,2,7},
{8, 0,8}}
```

```
(*join side-by-side as in Matlab [A B]*)
Join[a, b, 2]
```

```
{{8,5, 1, 9,0,1},
{8,10,8,10,2,7},
{6,0, 8, 8,0,8}}
```

```
(*join vertically as in Matlab [A;B]*)
Join[a, b, 1]
```

```
{{8, 5 , 1},
{8, 10, 8},
{6, 0 , 8},
{9, 0 , 1},
{10,2 , 7},
{8, 0 , 8}}
```

Matlab

```
A=rand(3,3)
B=rand(3,3)
[A B]
```

```
0.5472 0.2575 0.8143 0.3500 0.6160 0.8308
0.1386 0.8407 0.2435 0.1966 0.4733 0.5853
0.1493 0.2543 0.9293 0.2511 0.3517 0.5497
```

```
[A;B]
```

```
    0.5472    0.2575    0.8143
    0.1386    0.8407    0.2435
    0.1493    0.2543    0.9293
    0.3500    0.6160    0.8308
    0.1966    0.4733    0.5853
    0.2511    0.3517    0.5497
```

Maple

```
A:=ArrayTools:-RandomArray(3,3,distribution = uniform);
B:=ArrayTools:-RandomArray(3,3,distribution = uniform);
<A|B>;   #side by side

<A,B>;  #on top of each others

#or
Matrix([A,B]); #side by side
Matrix([[A],[B]]); #on top of each others
```

2.19 Copy the lower triangle to the upper triangle of a matrix to make symmetric matrix

Question posted on the net

```
please help me with simple to apply function that will construct
symmetric matrix from given just a half matrix with diagonal.
Eg:
```

```
From:
1  0  0  0
2  3  0  0
4  9  5  0
2  2  3  4

To give:

1  2  4  2
2  3  9  2
4  9  5  3
2  2  3  4
```

Many answers were given, below is my answer, and I also show how to do it in Matlab

Mathematica

```
a = {{1, 0, 0, 0},
     {2, 3, 0, 0},
     {4, 9, 5, 0},
     {2, 2, 3, 4}};

Transpose[LowerTriangularize[a, -1]] + a
```

```
{{1, 2, 4, 2},
 {2, 3, 9, 2},
 {4, 9, 5, 3},
 {2, 2, 3, 4}
}
```

Matlab

```
a =
     1     0     0     0
     2     3     0     0
     4     9     5     0
     2     2     3     4

a+tril(a,-1)'
```

```
ans =
     1     2     4     2
     2     3     9     2
     4     9     5     3
     2     2     3     4
```

Maple

```
A:=Matrix([[1,0,0,0],[2,3,0,0],[4,9,5,0],
B:=ArrayTools:-LowerTriangle(A,-1);
A:=A+B^%T;
```

```
[[1,2,4,2],
 [2,3,9,2],
 [4,9,5,3],
 [2,2,3,4]]
```

2.20 extract values from matrix given their index

Given a matrix A, and list of locations within the matrix, where each location is given by i, j entry, find the value in the matrix at these locations

Example, given

```
A={{1,2,3},
   {4,5,6},
   {7,8,9}}
```

obtain the entries at $1, 1$ and $3, 3$ which will be 1 and 9 in this example.

Mathematica

```
mat={{1,2,3},
     {4,5,6},
     {7,8,9}}

pos = { {1,1},{3,3}};
Map[ mat[[Sequence@@ # ]] & , pos ]
```

```
{1,9}
```

Another method (same really as above, but using Part explicit)

```
Map[Part[A, Sequence @@ #] &, pos]
```

```
{1,9}
```

Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9];
I=[1 3];  % setup the I,J indices
J=[1 3];
A(sub2ind(size(A),I,J))
```

```
ans =
     1     9
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
loc:=[[1,1],[3,3]];
map(x->A[op(x)],loc);
```

```
    [1, 9]
```

2.21 Convert N by M matrix to a row of length N M

Given

```
a=[1 2 3
   4 5 6
   7 8 9]
```

covert the matrix to one vector

```
[1 2 3 4 5 6 7 8 9]
```

Mathematica

```
a={{1,2,3},
   {4,5,6},
   {7,8,9}}

Flatten[a]
```

```
{1,2,3,4,5,6,7,8,9}
```

Matlab

```
a=[1 2 3;
   4 5 6;
   7 8 9]

a=a(:);
a'
```

```
 1 4 7 2 5 8 3 6 9
```

Maple Maple reshapes along columns, like Matlab. To get same result as Mathematica, we can transpose the matrix first. To get same result as Matlab, do not transpose.

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
v:=ListTools:-Flatten(convert(A,list));

v:=ListTools:-Flatten(convert(A^%T,list))
```

```
[1, 4, 7, 2, 5, 8, 3, 6, 9]

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2.22 find rows in a matrix based on values in different columns

Example, given Matrix

```
     1      9      0     10
     5      6      7      8
     3      9      2     10
```

Select rows which has 9 in the second column and 10 in the last column. Hence the result will be the first and third rows only

```
     1     9     0    10
     3     9     2    10
```

Mathematica

```
mat={{1,9,0,10},
      {5,6,7,8},
      {3,9,2,10}};

p = Position[mat[[All,{2,4}]],x_/;x===={9,10}]
```

```
{{1},{3}}
```

```
Extract[mat,p]
```

```
{{1,9,0,10},
 {3,9,2,10}}
```

Matlab

```
A=[1 9 0 10;
   5 6 7 8;
   3 9 2 10];

r=bsxfun(@eq,A(:,[2 4]),[9 10]);
ind=all(r,2);
A(ind,:)
```

```
ans =
     1     9     0    10
     3     9     2    10
```

Maple

```
restart;
A:=Matrix([[1,9,0,10],[5,6,7,8],[3,9,2,10]]);
nRow:=op([1,1],A);
<seq(`if`(A[i,2]=9 and A[i,-1]=10,A[i,..],NULL),i=1..nRow)>;

#OR
restart;
A:=Matrix([[1,9,0,10],[5,6,7,8],[3,9,2,10]]);
nRow:=LinearAlgebra:-RowDimension(A);
my_rows:=[seq(`if`(A[i,2]=9 and A[i,-1]=10,convert(A[i,..],list),NULL),i=1..nRow)];
Matrix(my_rows)
```

$$\begin{bmatrix} 1 & 9 & 0 & 10 \\ 3 & 9 & 2 & 10 \end{bmatrix}$$

There does not seem to be a direct way in Maple to take list of row Vectors, and use these to make a Matrix. This is why I had to convert to list in the above to make it work.

2.23 Select entries in one column based on a condition in another column

Given

```
A=[4  3
   6  4  ---->
   7  6  ---->
   2  1
   1  3
   9  2
   2  5  ---->
   1  2
   ]
```

Select elements in the first column only which has corresponding element in the second column greater than 3, hence the result will be

```
     [6 7 2]
```

Mathematica

```
mat = {{4, 3},
       {6, 4},
       {7, 6},
       {2, 1},
       {1, 3},
       {9, 2},
       {2, 5},
       {1, 2}};

r = Select[mat, #[[2]] > 3 &];
r[[All, 1]]
```

```
{6, 7, 2}
```

another way is to find the index using Position and then use Extract

```
loc = Position[mat, {x_, y_} /; y > 3]
r = Extract[mat, loc];
r[[All, 1]]
```

```
{6, 7, 2}
```

another way is to use Cases[]. This is the shortest way

```
Cases[mat, {x_, y_} /; y > 3 :> x]
```

```
{6, 7, 2}
```

Matlab

```
A=[4, 3;
   6, 4;
   7, 6;
   2, 1;
   1, 3;
   9, 2;
   2, 5;
   1, 2];

A(A(:,2)>3,1)
```

```
     6
     7
     2
```

Maple

In Maple, we must convert the Matrix to list of lists to work on each row at a time. I could not find a way to avoid this.

```
restart;
A:=<<4|3>,
    <6|4>,
    <7|6>,
    <2|1>,
    <1|3>,
    <9|2>,
    <2|5>,
    <1|2>>;
select(z->evalb(z[2]>3),convert(A,listlist))[..,1]
```

$$[6, 7, 2]$$

2.24 Locate rows in a matrix with column being a string

The problem is to select rows in a matrix based on string value in the first column. Then sum the total in the corresponding entries in the second column. Given. For example, given

```
mat = {'foobar', 77;
```

```
       'faabar', 81;
       'foobur', 22;
       'faabaa', 8;
       'faabian', 88;
       'foobar', 27;
       'fiijii', 52};
```

and given list

```
{'foo', 'faa'}
```

The problem is to select rows in which the string in the list is part of the string in the first column in the matrix, and then sum the total in the second column for each row found. Hence the result of the above should be

```
    'foo'    'faa'
    [126]    [ 177]
```

Mathematica

```
mat = {{"foobar", 77},
   {"faabar", 81},
   {"foobur", 22},
   {"faabaa", 8},
   {"faabian", 88},
   {"foobar", 27},
   {"fiijii", 52}};
lst = {"foo", "faa"};

foo[mat_, lst_] :=
  Select[mat, StringMatchQ[lst,
      StringTake[#[[1]], 3]] &];

r = Map[foo[mat, #] &, lst];

MapThread[{#1,
 Total[#2[[All, 2]]]}&,{lst, r}]
```

```
{{"foo", 126}, {"faa", 177}}
```

Matlab

```
A = {'foobar', 77;
   'faabar', 81;
   'foobur', 22;
   'faabaa', 8;
   'faabian', 88;
   'foobar', 27;
   'fiijii', 52};
lst= {'foo', 'faa'};
f=@(x) [x sum(cell2mat(A(strncmp(A(:,1),x,3),2)))];
r=arrayfun(@(i) f(lst(i)),1:2,'UniformOutput',false)
reshape([r{:}],2,2)
```

```
ans =
    'foo'    'faa'
    [126]    [177]
```

But notice that in Matlab, the answer is a cellarray. To access the numbers above

```
r{:}
r{1}{2}
r{2}{2}
```

```
ans =
    'foo'    [126]
ans =
    'faa'    [177]

ans =
   126

ans =
    177
```

2.25 Remove set of rows and columns from a matrix at once

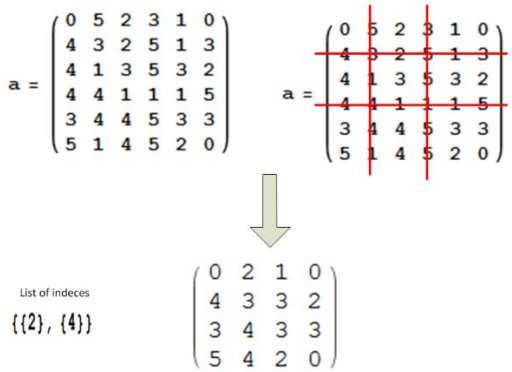
Given: square matrix, and list which represents the index of rows to be removed, and it also represents at the same time the index of the columns to be removed (it is square matrix, so only one list is needed).

output: the square matrix, with BOTH the rows and the columns in the list removed.

Assume valid list of indices.

This is an example: remove the second and fourth rows and the second and fourth

columns from a square matrix.



I asked this question at SO, and more methods are shown there at HTML

Mathematica Three methods are given.

method 1:

(credit for the idea to Mike Honeychurch at stackoverflow). It turns out it is easier to work with what we want to keep instead of what we want to delete so that Part[] can be used directly.

Hence, given a list of row numbers to remove, such as

```
pos = {2, 4};
```

Start by generating list of the rows and columns to keep by using the command Complement[], followed by using Part[]

```
a = {{0, 5, 2, 3, 1, 0},
     {4, 3, 2, 5, 1, 3},
     {4, 1, 3, 5, 3, 2},
     {4, 4, 1, 1, 1, 5},
     {3, 4, 4, 5, 3, 3},
     {5, 1, 4, 5, 2, 0}
   };
pos = {2, 4};
keep = Complement[Range[Length[a]], pos];
a[[keep, keep]]
```

Gives

```
{{0, 2, 1, 0},
```

```
 {4, 3, 3, 2},
 {3, 4, 3, 3},
 {5, 4, 2, 0}
}
```

Method 2: (due to Mr Wizard at stackoverflow)

```
ReplacePart[a, {{2},{4},{_, 2},{_, 4}}
      :> Sequence[]]
```

Gives

```
{{0, 2, 1, 0},
 {4, 3, 3, 2},
 {3, 4, 3, 3},
 {5, 4, 2, 0}
}
```

Method 3: (me)

use Pick. This works similar to Fortran pack(). Using a mask matrix, we set the entry in the mask to False for those elements we want removed. Hence this method is just a matter of making a mask matrix and then using it in the Pick[] command.

```
{nRow,nCol} = Dimensions[a];
mask = Table[True,{nRow},{nCol}];
pos  = {2,4};
mask[[All,pos]]=False;
mask[[pos,All]]=False;
r=Pick[a,mask]/.{}->Sequence[]
```

gives

```
Out[39]= {{0,2,1,0},
          {4,3,3,2},
          {3,4,3,3},
          {5,4,2,0}}
```

Matlab

```
a = [0, 5, 2, 3, 1, 0;
     4, 3, 2, 5, 1, 3;
     4, 1, 3, 5, 3, 2;
     4, 4, 1, 1, 1, 5;
     3, 4, 4, 5, 3, 3;
     5, 1, 4, 5, 2, 0
   ]

list=[2 4];
a(list,:)=[];
a(:,list)=[];
```

```
a =
     0     2     1     0
     4     3     3     2
     3     4     3     3
     5     4     2     0
```

Maple

```
A := Matrix([
     [0, 5, 2, 3, 1, 0],
     [4, 3, 2, 5, 1, 3],
     [4, 1, 3, 5, 3, 2],
     [4, 4, 1, 1, 1, 5],
     [3, 4, 4, 5, 3, 3],
     [5, 1, 4, 5, 2, 0]
   ]);
the_list:=[2,4];
A:=LinearAlgebra:-DeleteRow(A,the_list);
A:=LinearAlgebra:-DeleteColumn(A,the_list);
```

```
A:=[[0, 2, 1, 0],
    [4, 3, 3, 2],
    [3, 4, 3, 3],
    [5, 4, 2, 0]]
```

2.26 Convert list of separated numerical numbers to strings

Problem: given a list of numbers such as

```
{{1, 2},
 {5, 3, 7},
 {4}
```

```
}
```

convert the above to list of strings

```
{"12",
 "537",
 "4"}
```

Mathematica

```
a={{1,2},{5,3,7},{4}};
Map[ToString[#]&,a,{2}];
Map[StringJoin[#]&,%]
```

```
List["12","537","4"]
```

Matlab

```
a={[1,2],[5,3,7],[4]};
r=arrayfun(@(i) ...
   cellstr(num2str(a{i})),1:length(a));
r'
```

```
ans =
    '1  2'
    '5  3  7'
    '4'
```

answer below is due to Bruno Luong at Matlab newsgroup

```
a={[1,2],[5,3,7],[4]};
map='0':'9';
cellfun(@(x) map(x+1), a, 'uni', 0)
```

```
    '12'    '537'    '4'
```

Maple

```
a:=[[1,2],[5,3,7],[4]];
map(x->convert(cat(op(x)), string),a);
```

```
["12", "537", "4"]
```

2.27 Obtain elements that are common to two vectors

Given vector or list $d = [-9, 1, 3, -3, 50, 7, 19]$, $t = [0, 7, 2, 50]$, find the common elements.

Mathematica

```
d = {-9,1,3,-3,50,7,19};
t = {0,7,2,50};
Intersection[d,t]
```

```
Out[412]= {7,50}
```

Matlab

```
d=[-9 1 3 -3 50 7 19];
t =[0 7 2 50];
intersect(t,d)
```

```
ans =
     7    50
```

Maple

```
d := {-9,1,3,-3,50,7,19};
t := {0,7,2,50};
d intersect t;
```

```
{7, 50}
```

2.28 Sort each column (on its own) in a matrix

Given

```
    4     2     5
    2     7     9
   10     1     2
```

Sort each column on its own, so that the result is

```
    2     1     2
```

```
    4     2     5
   10     7     9
```

In Matlab, the sort command is used. But in the Mathematica, the Sort command is the same the Matlab's sortrows() command, hence it can't be used as is. Map is used with Sort to accomplish this.

Mathematica

```
mat={{4,2,5},
     {2,7,9},
     {10,1,2}};

Map[Sort[#]&,Transpose[mat]];
mat=Transpose[%]
```

```
{{2, 1, 2},
 {4, 2, 5},
 {10,7, 9}}
```

Matlab

```
A=[4  2 5;
   2  7 9;
   10 1 2];
sort(A,1)
```

```
    2     1     2
    4     2     5
   10     7     9
```

Maple

```
restart;
restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
map(x->sort(x),[LinearAlgebra:-Column(A,[1..-1])]);
Matrix(%);

#or, may simpler is

restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
for n from 1 to LinearAlgebra:-ColumnDimension(A)  do
    A[..,n]:=sort(A[..,n]);
od:
A
```

```
Matrix(3, 3, [[2, 1, 2],
              [4, 2, 5],
              [10, 7, 9]])
```

2.29 Sort each row (on its own) in a matrix

Given

```
    4     2     5
    2     7     9
   10     1     2
```

Sort each row on its own, so that the result is

```
    2     4     5
    2     7     9
    1     2     10
```

Mathematica

```
mat={{4, 2,5},
     {2, 7,9},
     {10,1,2}};
Map[Sort[#]&,mat]
```

```
{{2, 4, 5},
{2, 7, 9},
{1, 2, 10}}
```

Matlab

```
A=[4  2 5;
   2  7 9;
   10 1 2];
sort(A,2)
```

```
     2     4     5
     2     7     9
     1     2    10
```

Maple

```
restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
map(x->convert(sort(x),list),[LinearAlgebra:-Row(A,[1..-1])]);
Matrix(%);

#or, may simpler is

restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
for n from 1 to LinearAlgebra:-RowDimension(A)  do
    A[n,..]:=sort(A[n,..]);
od:
A
```

```
Matrix(3, 3, [[2, 4, 5],
              [2, 7, 9],
              [1, 2, 10]])
```

I had to convert each row to a list above for this to work. But for the case of columns (see last item earlier), this was not needed. Strange that one can't not construct a Matrix from list of Row vectors as with a list of Column vectors.

Maple 2021.

2.30 Sort a matrix row-wise using first column as key

Given

```
     4     2     5
     2     7     9
    10     1     2
```

Sort the matrix row-wise using first column as key so that the result is

```
    2     7     9
    4     2     5
   10     1     2
```

In Matlab, the sortrows() command is used. In Mathematica the Sort[] command is used as is. No need to do anything special.

Mathematica

```
mat={{4, 2, 5},
     {2, 7, 9},
     {10,1, 2}};
Sort[mat]
```

```
{{2,  7, 9},
{4,  2, 5},
{10, 1, 2}}
```

Matlab

```
A=[4  2 5;
   2  7 9;
   10 1 2];
sortrows(A)
```

```
     2     7     9
     4     2     5
    10     1     2
```

Maple

```
restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
convert(A,listlist);
sort(%);
Matrix(%)
```

```
Matrix(3, 3, [[2, 7, 9],
              [4, 2, 5],
              [10, 1, 2]])
```

In Maple 2023 we can now do the following to get the same output

```
restart;
A:=Matrix([[4,2,5],[2,7,9],[10,1,2]]);
ArrayTools:-SortBy( A, 'column', 1 );
```

```
Matrix(3, 3, [[2, 7, 9],
              [4, 2, 5],
              [10, 1, 2]])
```

2.31 Sort a matrix row-wise using non-first column as key

Given

```
    4     2     5
    2     7     9
   10     1     2
```

Sort the matrix row-wise using the second column as key so that the result is

```
   10     1     2
    4     2     5
    2     7     9
```

In Matlab, the sortrows() command is used, but now we tell it to use the second column as key.

In Mathematica the SortBy[] command is now used but we tell it to use the second slot as key.

Mathematica

```
mat={{4, 2, 5},
     {2, 7, 9},
     {10,1, 2}};
(*sort by second element as key*)
SortBy[mat,#[[2]]&]
```

```
{{10, 1, 2},
{4,  2, 5},
{2,  7, 9}}
```

Matlab

```
A=[4  2 5;
   2  7 9;
   10 1 2];
sortrows(A,2)
```

```
    10     1     2
     4     2     5
     2     7     9
```

Maple

```
restart;
A:=Matrix([[4,2,5],
           [2,7,9],
           [10,1,2]]);
convert(A,listlist);
sort(%,key=(x->x[2]));
Matrix(%)
```

```
Matrix(3, 3, [[10, 1, 2],
              [4, 2, 5],
              [2, 7, 9]])
```

In Maple 2023 we can now do the following to get the same output

```
restart;
A:=Matrix([[4,2,5],
           [2,7,9],
           [10,1,2]]);
ArrayTools:-SortBy( A, 'column', 2 );
```

```
Matrix(3, 3, [[10, 1, 2],
              [4, 2, 5],
              [2, 7, 9]])
```

2.32 Replace the first nonzero element in each row in a matrix by some value

Problem: Given a matrix, replace the first nonzero element in each row by some a specific value. This is an example. Given matrix A below, replace the first non-zero element in each row by -1, then

$$A = \begin{pmatrix} 50 & 75 & 0 \\ 50 & 0 & 100 \\ 0 & 75 & 100 \\ 75 & 100 & 0 \\ 0 & 75 & 100 \\ 0 & 75 & 100 \end{pmatrix} \text{ will become } B = \begin{pmatrix} -1 & 75 & 0 \\ -1 & 0 & 100 \\ 0 & -1 & 100 \\ -1 & 100 & 0 \\ 0 & -1 & 100 \\ 0 & -1 & 100 \end{pmatrix}$$

Mathematica

```
A={ {50, 75,  0},{50, 0,   100},
    {0,  75,  100},{75, 100, 0},
    {0,  75,  100},{0,  75,  100}};
ReplacePart[A,DeleteDuplicates[
    Position[A,_?(#!=0&)],
      (#1[[1]]==#2[[1]]&)]->-1]
```

Solution due to Bob Hanlon (from Mathematica newsgroup)

```
r=Union[Position[A,_?(# !=0&)],
  SameTest->(#1[[1]]==#2[[1]]&)];
ReplacePart[A,r->-1]
```

Solution by Fred Simons (from Mathematica newsgroup)

```
r=Split[Position[A,_?(# !=0&),{2}],
      #1[[1]]==#2[[1]]&][[All,1]];
ReplacePart[A,r->-1]
```

Solution due to Adriano Pascoletti (from Mathematica newsgroup)

```
r=(First /@ Split[#1,First[#1]===First[#2]&]&)
       [Position[A,x_/;x!=0]]
ReplacePart[A,r->-1]
```

Solution due to Oliver Ruebenkoenig (from Mathematica newsgroup)

```
r=First /@ SplitBy[ Drop[ArrayRules[
     SparseArray[A]][[All,1]],-1],First]
ReplacePart[A,r->-1]
```

Solution due to Szabolcs Horvát (from Mathematica newsgroup)

```
r=MapIndexed[{First[#2],1+LengthWhile[#1,#==0&]}&,A]
ReplacePart[A,r->-1]
```

These all give

```
{{-1,75,0},{-1,0,100},{0,-1,100},
{-1,100,0},{0,-1,100},{0,-1,100}}
```

Matlab

```
A=[50 75 0;
   50 0 100;
   0 75 100;
   75 100 0;
   0 75 100;
   0 75 100];

[I,J] = ind2sub(size(A),find(A~=0));
[b,c] = unique(I,'first');
A(sub2ind(size(A),b,J(c)))=-1
```

This solution due to Bruno Luong (from matlab newsgroup)

```
B = cumsum(A~=0,2)>0
B = [false(size(B,1),1) B]
A(logical(diff(B,1,2))) = -1
```

This solution due to Jos from matlab newsgroup

```
A((cumsum(~~A,2)==1) & (A ~= 0)) = -1
```

The above solutions all give

```
A =
    -1    75     0
    -1     0   100
     0    -1   100
    -1   100     0
     0    -1   100
     0    -1   100
```

Maple

```
restart;
A:=Matrix([[50,75,0],
           [50,0,100],
           [0,75,100],
           [75,100,0],
           [0,75,100],
           [0,75,100]]);
for n from 1 to LinearAlgebra:-RowDimensi
    idx:=ArrayTools:-SearchArray(A[n,..],
    if numelems(idx)<>0 then
       A[n,idx[1]]:=-1;
    fi;
 od:
A
```

```
Matrix(6, 3, [[-1, 75, 0],
              [-1, 0, 100],
              [0, -1, 100],
              [-1, 100, 0],
              [0, -1, 100],
              [0, -1, 100]])
```

2.33 Perform outer product and outer sum between two vector

Problem: Given 2 vectors, perform outer product and outer sum between them. The outer operation takes the first element in one vector and performs this operation on each element in the second vector. This results in first row. This is repeated for each of the elements in the first vector. The operation to perform can be any valid operation on these elements.

Mathematica

using symbolic vectors. Outer product

```
Remove["Global`*"]

v1={a,b,c};
v2={e,f,g};
Outer[Times,v1,v2]
```

```
{{a e, a f, a g},
{b e, b f, b g},
{c e, c f, c g}}
```

Outer sum

```
Outer[Plus,v1,v2]
```

```
{{a+e, a+f, a+g},
{b+e, b+f, b+g},
{c+e, c+f, c+g}}
```

using numerical vectors. Outer product

```
v1={1,2,3};
v2={4,5,6};
Outer[Times,v1,v2]
```

```
{{4,  5,  6},
{8,  10, 12},
{12, 15, 18}}
```

Outer sum

```
Outer[Plus,v1,v2]
```

```
  {{5, 6, 7},
   {6, 7, 8},
   {7, 8, 9}}
```

Matlab

Outer product

```
v1=[1 2 3];
v2=[4 5 6];

v1'*v2
```

```
ans =
     4     5     6
     8    10    12
    12    15    18
```

Outer sum

```
v1=[1 2 3];
v2=[4 5 6];
meshgrid(v1)+meshgrid(v2)'
```

```
ans =
     5     6     7
     6     7     8
     7     8     9
```

Maple

Due to Carl Love from the Maple newsgroup

```
restart;

Outer:= proc(OP, L1, L2)
    local a,b;
   [seq](seq(OP(a,b), b in L2), a in L1)
end proc:

L1:=[a,b,c];
L2:=[d,e,f];
Outer(`*`,L1,L2);
```

```
[a*d,a*e,a*f,b*d,
  b*e, b*f, c*d, c*e, c*f]
```

```
Outer(`+`,L1,L2);
```

```
[a+d, a+e, a+f, b+d,
  b+e, b+f, c+d, c+e, c+f]
```

2.34 Find the rank and the bases of the Null space for a matrix A

Problem: Find the rank and nullities of the following matrices, and find the bases of the range space and the Null space.

$$A = \begin{pmatrix} 2 & 3 & 3 & 4 \\ 0 & -1 & -2 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Mathematica

```
mat={{2,3,3,4},
     {0,-1,-2,2},
     {0,0,0,1}};

{nRow,nCol} = Dimensions[mat]
```

3,4

```
Print["Rank (or dimension of the range space)=",
       MatrixRank[mat]]
```

```
Rank (or dimension of the range space)=3
```

```
Print["Dimension of the Null Space=",
      nCol-MatrixRank[mat]]
```

```
Dimension of the Null Space=1
```

```
Print["Basis for Null Space=",NullSpace[mat]]
```

```
Basis for Null Space={{3,-4,2,0}}
```

Matlab

```
A=[2 3 3 4;
   0 -1 -2 2;
   0 0 0 1]

[nRow,nCol]=size(A);

r = rank(A);
fprintf('A range space dimension=%d\n',r);
fprintf('A null space dimension= %d\n',nCol-r);
fprintf('Basic for null space of A =');
null(A,'r')'
```

```
A range space dimension=3
A null space dimension= 1
Basic for null space of A =

ans =
 1.5000   -2.0000  1.0000  0
```

Maple

```
restart;
A:=Matrix([[2,3,3,4],[0,-1,-2,2],[0,0,0,1]]);
LinearAlgebra:-Rank(A);
LinearAlgebra:-ColumnDimension(A)-LinearAlgebra:-
LinearAlgebra:-NullSpace(A)
```

```
3
1
  [3/2
   -2
   1
   0]
```

2.35 Find the singular value decomposition (SVD) of a matrix

Problem: Find the SVD for the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ Notice that in Maple, the singular values matrix, normally called S, is returned as a column vector. So need to call DiagonalMatrix() to format it as expected.

Mathematica

```
mat = {{1,2,3},
       {4,5,6}}

{u,s,v}=N[SingularValueDecomposition[mat]]
```

```
{{0.386318,-0.922366},
{0.922366,0.386318}}

{{9.50803,0.,0.},
{0.,0.77287,0.}}

{{0.428667,0.805964,0.408248},
{0.566307,0.112382,-0.816497},
{0.703947,-0.581199,0.408248}}
```

```
(*Reconstruct A from its components *)
u.s.Transpose[v]
```

```
{{1.,2.,3.},
{4.,5.,6.}}
```

Matlab

```
A=[1 2 3;
   4 5 6];
[u,s,v]=svd(A);
u
s
v
```

```
u =
   -0.3863   -0.9224
   -0.9224    0.3863

s =

    9.5080         0         0
         0    0.7729         0

v =

   -0.4287    0.8060    0.4082
   -0.5663    0.1124   -0.8165
   -0.7039   -0.5812    0.4082
```

```
u*s*v'
```

```
ans =
    1.0000    2.0000    3.0000
    4.0000    5.0000    6.0000
```

Maple

```
restart;
with(LinearAlgebra):
A:=Matrix([[1,2,3],[4,5,6.]]);
```

```
     [1    2    3 ]
A := [            ]
     [4    5    6.]
```

```
m,n:=Dimensions(A):
u,s,v:=SingularValues(A,
    output=['U', 'S', 'Vt']):
u;
```

```
 [0.386317703118612    -0.922365780077058]
 [                                       ]
 [0.922365780077058    0.386317703118612 ]
```

```
s:=DiagonalMatrix(s,m,n);
v;
```

```
     [9.50803200069572            0
s := [
     [       0            0.772869635673485

     [0.428667133548626    0.566306918848035
v:=  [0.805963908589298    0.112382414096594
     [0.408248290463863    -0.816496580927726
```

```
> s2:=DiagonalMatrix(SingularValues(A,
                        output=['S']),m,n);
```

```
      [9.50803200069572        0            0
s2 := [
      [       0         0.772869635673485
```

```
> evalf(u.s.v);
```

```
[1.    2.    3.]
[              ]
[4.    5.    6.]
```

2.36 Solve $Ax = b$

Solve for x in the following system of equations

```
[ 1 2 3 ][x1]   [ 1 ]
[ 7 5 6 ][x2]=  [ 2 ]
[ 7 8 9 ][x3]   [ 3 ]
```

Mathematica

```
mat={{1,2,3},
      {7,5,6},
      {7,8,9}};
b = {1,2,3}
LinearSolve[mat,b]
```

```
{0,0,1/3}
```

Matlab

```
format long
A=[1 2 3;
   7 5 6;
   7 8 9];
b=[1;2; 3];
A\b
```

```
ans =
                    0
                    0
   0.333333333333333
```

Maple

```
restart;
A:=Matrix([[1,2,3],[7,5,6],[7,8,9]]);
b:=Vector([1,2,3]);
LinearAlgebra:-LinearSolve(A,b)
```

```
  [0,
   0,
   1/3]
```

Fortran

```
 program t2
implicit none
integer, parameter :: N=3
real(8),   DIMENSION(N, N) :: &
           A=DBLE(reshape([ 1.0, 2.0, 3.0,&
                       7.0, 5.0, 6.0,&
                       7.0, 8.0, 9.0], shape(A), order=[2,1]))

real(8), parameter, DIMENSION(N, 1) :: &
           b=DBLE(reshape([1.0, 2.0, 3.0], shape(b)))

real(8), DIMENSION(N, 1) :: IPIV
integer :: INFO

 CALL DGETRF( N, N, A, N, IPIV, INFO )
 if (INFO .eq. 0 ) then
     CALL DGETRS( 'N', N,1, A, N, IPIV, b , N, INFO)
     if ( INFO .eq. 0 ) then
        print *, 'solution is',b
     else
        print *, 'failed DGETRS, INFO=',INFO
     end if
 else
     print *, 'failed DGETRF, INFO=',INFO
 end if
end program
```

compile and run

```
$ gfortran -std=f2003 -Wextra -Wall -pedantic -funroll-loops
  -ftree-vectorize -march=native  -Wsurprising -Wconversion
  t2.f90  /usr/lib/liblapack.a /usr/lib/libblas.a

$ ./a.exe
 solution is
 -5.28677630773884192E-018 -3.70074341541718826E-017  0.33333333333333337
```

2.37 Find all nonzero elements in a matrix

Given a matrix, find the locations and the values of all nonzero elements. Hence given the matrix

$$\begin{pmatrix} 0 & 0 & 1 \\ 10 & 0 & 2 \\ 3 & 0 & 0 \end{pmatrix}$$

the positions returned will be $(1,3),(2,1),(2,3),(3,1)$ and the corresponding values are $1,10,2,3$.

Mathematica

In Mathematica, standard Mathematica matrix operations can be used, or the matrix can be converted to SparseArray and special named operation can be used on it.

```
mat  = {{0 , 0, 1},
        {10, 0, 2},
        {3 , 4, 0}};

sp  = SparseArray[mat];
pos = sp["NonzeroPositions"]
```

```
{{1,3},{2,1},{2,3},{3,1},{3,2}}
```

```
values=sp["NonzeroValues"]
```

```
{1,10,2,3,4}
```

Or standard list operations can be used

```
mat  = {{0 , 0, 1},
        {10, 0, 2},
        {3 , 4, 0}};

(*find values not zero *)

Cases[mat,x_ /;Not[PossibleZeroQ[x]]:>x,2]
```

```
{1,10,2,3,4}
```

```
(*find the index *)

Position[mat,x_/;x!=0]
```

```
{{1,3},{2,1},{2,3},{3,1},{3,2}}
```

Matlab

```
A=[0 0 1; 10 0 2; 3 4 0];
values= nonzeros(A)
```

```
values =
    10
     3
     4
     1
     2
```

```
%find locations
[I,J]=find(A)
```

```
I =
     2
     3
     3
     1
     2
J =
     1
     1
     2
     3
     3
```

Maple

```
A:=Matrix([[0,0,1],[10,0,2],[3,4,0]]);
row, col:=ArrayTools:-SearchArray(A);

#values
map(n->A[row[n],col[n]],[$numelems(row)])
```

```
row, col := Vector[column](5, [2, 3, 3, 1, 2]),
            Vector[column](5, [1, 1, 2, 3, 3])


[10, 3, 4, 1, 2]
```

2.38 evaluate $f(x)$ on a vector of values

Given a function $f(x)$ evaluate it for each value contained in a vector. For example, given $f(x) = x^2$ evaluate it on $(1, 2, 3)$ such that the result is $(1, 4, 9)$.

Mathematica

```
Clear[f,x]
f[x_]:=x^2
v={1,2,3};
f[v]
```

```
{1,4,9}
```

Matlab

```
clear all;
v=[1 2 3]
f=@(x) x.^2
f(v)
```

```
ans =
     1     4     9
```

Maple

```
map(x->x^2,[1,2,3])
```

```
[1, 4, 9]
```

2.39 generates equally spaced N points between x_1 and x_2

Mathematica

```
x1=1;
x2=3;
FindDivisions[{x1,x2},5]
```

```
Out[48]= {1,3/2,2,5/2,3}
N[%]
Out[49]= {1.,1.5,2.,2.5,3.}
```

Matlab

```
clear all;
x1 = 1;
x2 = 3;
linspace(x1,x2,5)
```

```
ans =
 1.0000   1.5000   2.0000   2.5000   3.0000
```

Maple

```
x1 := 1; x2 := 3;
ArrayTools:-RegularArray(x1..x2,5)
```

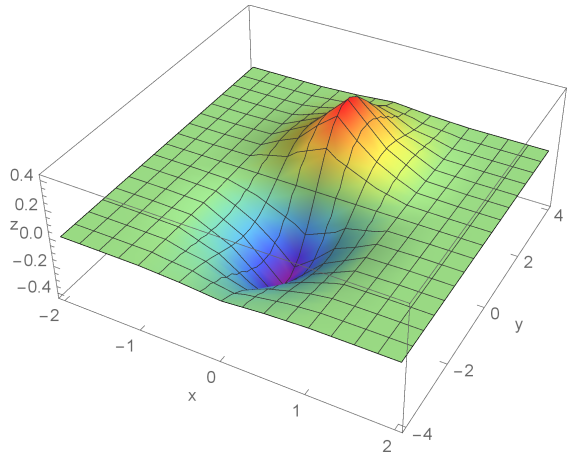
```
Vector[row](5, [1.0000, 1.5000, 2.0000, 2.5000, 3
```

2.40 evaluate and plot a f(x,y) on 2D grid of coordinates

Evaluate $x \exp^{-x^2-y^2}$ on 2D cartesian grid between $x = -2 \cdots 2$ and $y = -4 \cdots 4$ using $h = 0.2$ for grid spacing.

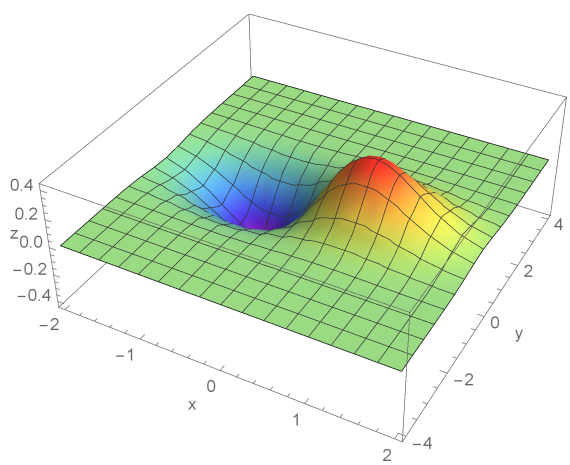
Mathematica

```
f[x_,y_]:=x Exp[-x^2-y^2];
data = Table[f[x,y],{x,-2,2,.2},{y,-4,4,.2}];
ListPlot3D[data,
 PlotRange->All,
 ColorFunction->"Rainbow",
 AxesLabel->{"x","y","z"},
 LabelStyle->12,
 Ticks->{Range[-2,2,1],Range[-4,4,1],Automatic},
 DataRange->{{-2,2},{-4,4},Automatic},
 ImageSize->400]
```



The above can also be done using Plot3D

```
f[x_,y_]:=x Exp[-x^2-y^2];
Plot3D[f[x,y],{x,-2,2},{y,-4,4},
 PlotRange->All,
 ColorFunction->"Rainbow",
 AxesLabel->{"x","y","z"},
 LabelStyle->12,
 ImageSize->400]
```

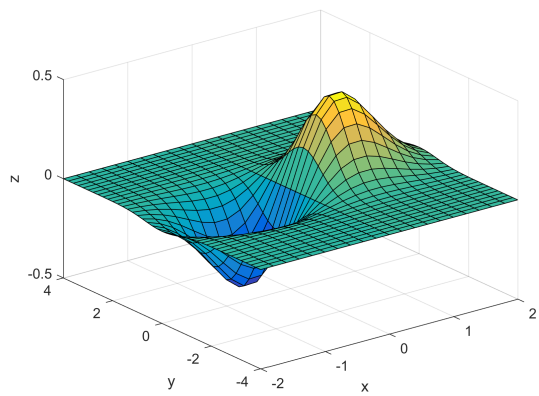


I need to sort out the orientation difference between the two plots above.

Matlab

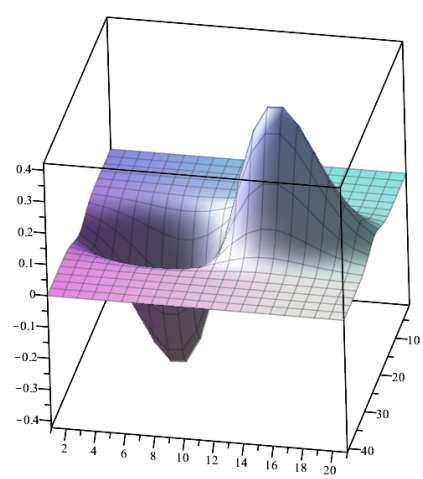
```
[X,Y] = meshgrid(-2:.2:2, -4:.2:4);
 Z = X .* exp(-X.^2 - Y.^2);
```

```
surf(X,Y,Z)
xlabel('x');
ylabel('y');
zlabel('z');
```



Maple

```
f:=(x,y)->x*exp(-x^2-y^2);
data:=[seq([seq(f(x,y),x=-2..2,.2)],y=-4..4,.2)]:
plots:-listplot3d(data)
```



2.41 Find determinant of matrix

Given a square matrix, find its determinant. In Mathematica, the Det[] command is used. In Matlab the det() command is used.

Mathematica

```
mat=Table[RandomReal[],{3},{3}]
Det[mat]
```

```
-0.317792605942287
```

Matlab

```
A=rand(3,3);
det(A)
```

```
-0.276653966272994
```

Maple

```
A:=LinearAlgebra:-RandomMatrix(3,3,generat
LinearAlgebra:-Determinant(A)
```

```
0.4101995534
```

2.42 Generate sparse matrix with n by n matrix repeated on its diagonal

Given matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$, generate the following sparse matrix with this matrix on the diagonal

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 8 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 & 8 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 8 & 9 \end{pmatrix}$$

Mathematica

```
mat = N[{{1,2,3},{4,5,6},{7,8,9}}];
sp  = SparseArray[Band[{1,1}]->
        ConstantArray[mat,{3}]]
MatrixForm[sp]
```

$$\begin{pmatrix} 1. & 2. & 3. & 0 & 0 & 0 & 0 & 0 & 0 \\ 4. & 5. & 6. & 0 & 0 & 0 & 0 & 0 & 0 \\ 7. & 8. & 9. & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1. & 2. & 3. & 0 & 0 & 0 \\ 0 & 0 & 0 & 4. & 5. & 6. & 0 & 0 & 0 \\ 0 & 0 & 0 & 7. & 8. & 9. & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1. & 2. & 3. \\ 0 & 0 & 0 & 0 & 0 & 0 & 4. & 5. & 6. \\ 0 & 0 & 0 & 0 & 0 & 0 & 7. & 8. & 9. \end{pmatrix}$$

Matlab

```
A  = [1 2 3;
      4 5 6;
      7 8 9];
Iy = speye(3);
full(Iy)
```

```
ans =
     1     0     0
     0     1     0
     0     0     1
```

```
sp = kron(Iy,A);
full(sp)
```

```
ans =
 1  2  3  0  0  0  0  0  0
 4  5  6  0  0  0  0  0  0
 7  8  9  0  0  0  0  0  0
 0  0  0  1  2  3  0  0  0
 0  0  0  4  5  6  0  0  0
 0  0  0  7  8  9  0  0  0
 0  0  0  0  0  0  1  2  3
 0  0  0  0  0  0  4  5  6
 0  0  0  0  0  0  7  8  9
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
Iy:=LinearAlgebra:-IdentityMatrix(3);
LinearAlgebra:-KroneckerProduct(Iy,A)
```

```
Matrix(9, 9, [[1, 2, 3, 0, 0, 0, 0, 0, 0],
              [4, 5, 6, 0, 0, 0, 0, 0, 0],
              [7, 8, 9, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 1, 2, 3, 0, 0, 0],
              [0, 0, 0, 4, 5, 6, 0, 0, 0],
              [0, 0, 0, 7, 8, 9, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 1, 2, 3],
              [0, 0, 0, 0, 0, 0, 4, 5, 6],
              [0, 0, 0, 0, 0, 0, 7, 8, 9]])
```

2.43 Generate sparse matrix for the tridiagonal representation of second difference operator in 1D

The second derivative $\frac{d^2u}{dx^2}$ is approximated by $\frac{u_{i-1}-2u_i+u_{i+1}}{h^2}$ where h is the grid spacing. Generate the A matrix that represent this operator for $n = 4$ where n is the number of internal grid points on the line

Mathematica

```
numberOfInternalGridPoints = 4;
n  = 3*internalGridPoints;
sp = SparseArray[{Band[{1,1}]->-2,
                  Band[{1,2}]->1,
                  Band[{2,1}]->1},
                  {n,n}]
Out[103]= SparseArray[<34>,{12,12}]
MatrixForm[sp]
```

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Matlab

```
internalGridPoints = 4;
n = 3*internalGridPoints;
e = ones(n,1);
sp = spdiags([e -2*e e], -1:1, n,n);
full(sp)
```

```
ans =
-2  1  0  0  0  0  0  0  0  0  0   0
 1 -2  1  0  0  0  0  0  0  0  0   0
 0  1 -2  1  0  0  0  0  0  0  0   0
 0  0  1 -2  1  0  0  0  0  0  0   0
 0  0  0  1 -2  1  0  0  0  0  0   0
 0  0  0  0  1 -2  1  0  0  0  0   0
 0  0  0  0  0  1 -2  1  0  0  0   0
 0  0  0  0  0  0  1 -2  1  0  0   0
 0  0  0  0  0  0  0  1 -2  1  0   0
 0  0  0  0  0  0  0  0  1 -2  1   0
 0  0  0  0  0  0  0  0  0  1 -2   1
 0  0  0  0  0  0  0  0  0  0  1  -2
```

Maple

```
interface(rtablesize=16):
LinearAlgebra:-BandMatrix([1,-2,1],1,12,12);
```

$$\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

2.44 Generate sparse matrix for the Laplacian differential operator $\nabla^2 u$ for 2D grid

$\nabla^2 u = f$ in 2D is defined as $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$ and the Laplacian operator using second order standard differences results in $\left(\frac{u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}-4u_{i,j}}{h^2}\right) = f_{i,j}$ where h is the grid size. The above is solved for x in the form $Ax = f$ by generating the A matrix and taking into consideration the boundary conditions. The follows show how to generate the sparse representation of A. Assuming the number of unknowns $n = 3$ in one direction, there are 9 unknowns to solve for and the A matrix will be 9 by 9.

Matlab

```
internalPoints=3;
e   = ones(internalPoints,1);
spe = spdiags([e -2*e e], -1:1,...
   internalPoints,internalPoints);
Iz  = speye(internalPoints);
sp  = kron(Iz,spe)+kron(spe,Iz);
full(sp)
```

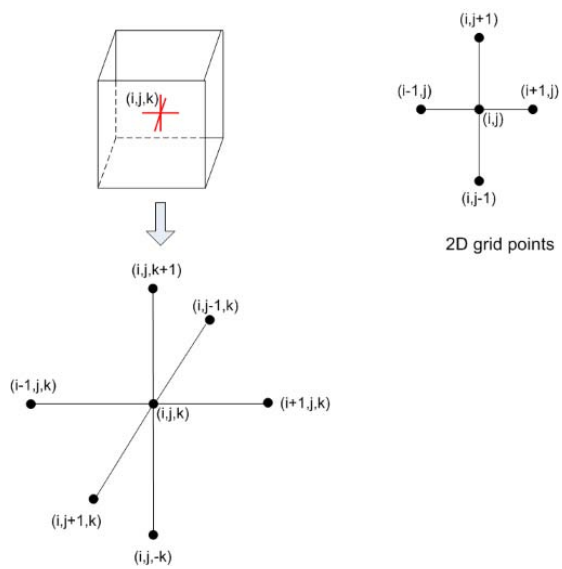
```
ans =
 -4  1  0  1  0  0  0  0  0
  1 -4  1  0  1  0  0  0  0
  0  1 -4  0  0  1  0  0  0
  1  0  0 -4  1  0  1  0  0
  0  1  0  1 -4  1  0  1  0
  0  0  1  0  1 -4  0  0  1
  0  0  0  1  0  0 -4  1  0
  0  0  0  0  1  0  1 -4  1
  0  0  0  0  0  1  0  1 -4
```

2.45 Generate sparse matrix for the Laplacian differential operator for 3D grid

The goal is to solve

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=-f(x,y,z)$$

On the unit cube. The following diagram was made to help setting up the 3D scheme to approximate the above PDE



The discrete approximation to the Laplacian in 3D is

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=\frac{1}{h^2}(U_{i-1,j,k}+U_{i+1,j,k}+U_{i,j-1,k}+U_{i,j+1,k}+U_{i,k,k-1}+U_{i,j,k+1}-6U_{i,j,k})$$

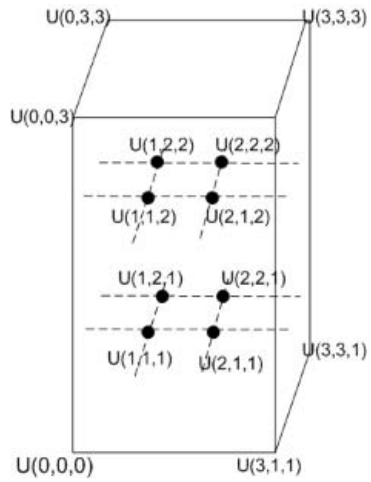
For the direct solver, the A matrix needs to be formulated. From

$$\frac{1}{h^2}(U_{i-1,j,k}+U_{i+1,j,k}+U_{i,j-1,k}+U_{i,j+1,k}+U_{i,k,k-1}+U_{i,j,k+1}-6U_{i,j,k})=f_{i,j,k}$$

Solving for $U_{i,j,k}$ results in

$$U_{i,j,k}=\frac{1}{6}\left(U_{i-1,j,k}+U_{i+1,j,k}+U_{i,j-1,k}+U_{i,j+1,k}+U_{i,k,k-1}+U_{i,j,k+1}-h^2 f_{i,j,k}\right)$$

To help make the A matrix, a small example with $n=2$, is made. The following diagram uses the standard numbering on each node

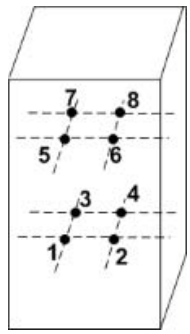


By traversing the grid, left to right, then inwards into the paper, then upwards, the following A matrix results

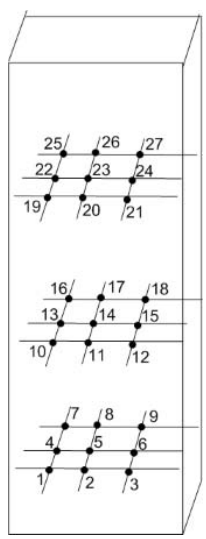
$$\begin{pmatrix} -6 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -6 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -6 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & -6 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -6 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & -6 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & -6 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & -6 \end{pmatrix} \begin{pmatrix} U_{1,1,1} \\ U_{2,1,1} \\ U_{1,2,1} \\ U_{2,2,1} \\ U_{1,1,2} \\ U_{2,1,2} \\ U_{1,2,2} \\ U_{2,2,2} \end{pmatrix} = h^3 \begin{pmatrix} f_{1,1,1} \\ f_{2,1,1} \\ f_{1,2,1} \\ f_{2,2,1} \\ f_{1,1,2} \\ f_{2,1,2} \\ f_{1,2,2} \\ f_{2,2,2} \end{pmatrix}$$

The recursive pattern involved in these A matrices can now be seen. Each A matrix contains inside it a block on its diagonal which repeats n times. Each block in turn contain inside it, on its diagonal, smaller block, which also repeats n times.

It is easier to see the pattern of building A by using numbers for the grid points, and label them in the same order as they would be visited, this allowes seeing the connection between each grid point to the other easier. For example, for $n = 2$,



The connections are now more easily seen. Grid point 1 has connection to only points $2, 3, 5$. This means when looking at the A matrix, there will be a 1 in the first row, at columns $2, 3, 5$. Similarly, point 2 has connections only to $1, 4, 6$, which means in the second row there will be a 1 at columns $1, 4, 6$. Extending the number of points to $n = 3$ to better see the pattern of A results in



From the above it is seen that for example point 1 is connected only to $2, 4, 10$ and point 2 is connected to $1, 3, 5, 11$ and so on.

The above shows that each point will have a connection to a point which is numbered n^2 higher than the grid point itself. n^2 is the size of the grid at each surface. Hence, the general A matrix, for the above example, can now be written as

$$
\begin{pmatrix}
-6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & -6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -6 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -6 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -6 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -6
\end{pmatrix}
\begin{pmatrix}
U_{111} \\ U_{211} \\ U_{311} \\ U_{121} \\ U_{221} \\ U_{321} \\ U_{131} \\ U_{231} \\ U_{331} \\ U_{112} \\ U_{212} \\ U_{312} \\ U_{122} \\ U_{222} \\ U_{322} \\ U_{132} \\ U_{232} \\ U_{332} \\ U_{113} \\ U_{213} \\ U_{313} \\ U_{123} \\ U_{223} \\ U_{323} \\ U_{133} \\ U_{233} \\ U_{333}
\end{pmatrix}
= h^2
\begin{pmatrix}
f_{111} \\ f_{211} \\ f_{311} \\ f_{121} \\ f_{221} \\ f_{321} \\ f_{131} \\ f_{231} \\ f_{331} \\ f_{112} \\ f_{212} \\ f_{312} \\ f_{122} \\ f_{222} \\ f_{322} \\ f_{132} \\ f_{232} \\ f_{332} \\ f_{113} \\ f_{213} \\ f_{313} \\ f_{123} \\ f_{223} \\ f_{323} \\ f_{133} \\ f_{233} \\ f_{333}
\end{pmatrix}
$$

The recursive structure can be seen. There are $n = 3$ main repeating blocks on the diagonal, and each one of them in turn has $n = 3$ repeating blocks on its own diagonal. Here $n = 3$, the number of grid points along one dimension.

Now that the A structure is understood, the A matrix can be generated.

Example 1: Using $n_x = 2$, $n_y = 2$, $n_z = 2$. These are the number of grid points in the x, y, z directions respectively.

Matlab

```
nx=2;
ny=2;
ex=ones(nx,1);
Lx=spdiags([ex -3*ex ex],[-1 0 1],nx,nx);
ey=ones(ny,1);
Ly=spdiags([ey -3*ey ey],[-1 0 1],ny,ny);

Ix=speye(nx);
Iy=speye(ny);
L2=kron(Iy,Lx)+kron(Ly,Ix);

nz=2;
N=nx*ny*nz;
e=ones(N,1);
L=spdiags([e e],[-nx*ny nx*ny],N,N);
Iz=speye(nz);

A=kron(Iz,L2)+L;
full(A)
```

```
ans =
 -6  1  1  0  1  0  0  0
  1 -6  0  1  0  1  0  0
  1  0 -6  1  0  0  1  0
  0  1  1 -6  0  0  0  1
  1  0  0  0 -6  1  1  0
  0  1  0  0  1 -6  0  1
  0  0  1  0  1  0 -6  1
  0  0  0  1  0  1  1 -6
```

Example 2: Using $n_x = 2$, $n_y = 2$, $n_z = 3$.

Matlab

```
nx=2;
ny=2;
ex=ones(nx,1);
Lx=spdiags([ex -3*ex ex],[-1 0 1],nx,nx);
ey=ones(ny,1);
Ly=spdiags([ey -3*ey ey],[-1 0 1],ny,ny);

Ix=speye(nx);
Iy=speye(ny);
L2=kron(Iy,Lx)+kron(Ly,Ix);

nz=3;
N=nx*ny*nz;
e=ones(N,1);
L=spdiags([e e],[-nx*ny nx*ny],N,N);
Iz=speye(nz);

A=kron(Iz,L2)+L;
full(A)
```

```
ans =
 -6  1  1  0  1  0  0  0  0  0  0  0
  1 -6  0  1  0  1  0  0  0  0  0  0
  1  0 -6  1  0  0  1  0  0  0  0  0
  0  1  1 -6  0  0  0  1  0  0  0  0
  1  0  0  0 -6  1  1  0  1  0  0  0
  0  1  0  0  1 -6  0  1  0  1  0  0
  0  0  1  0  1  0 -6  1  0  0  1  0
  0  0  0  1  0  1  1 -6  0  0  0  1
  0  0  0  0  1  0  0  0 -6  1  1  0
  0  0  0  0  0  1  0  0  1 -6  0  1
  0  0  0  0  0  0  1  0  1  0 -6  1
  0  0  0  0  0  0  0  1  0  1  1 -6
```

Example 3: Using $n_x = 3$, $n_y = 3$, $n_z = 3$.

Matlab

```
nx=3;
ny=3;
ex=ones(nx,1);
Lx=spdiags([ex -3*ex ex],[-1 0 1],nx,nx);
ey=ones(ny,1);
Ly=spdiags([ey -3*ey ey],[-1 0 1],ny,ny);

Ix=speye(nx);
Iy=speye(ny);
L2=kron(Iy,Lx)+kron(Ly,Ix);

nz=3;
N=nx*ny*nz;
e=ones(N,1);
L=spdiags([e e],[-nx*ny nx*ny],N,N);
Iz=speye(nz);

A=kron(Iz,L2)+L;
full(A)
```

```
>> full(A)

ans =

    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1    -6     0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     1     0     0    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     1    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     1    -6     0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     0     0    -6     1     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     1    -6     1     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     1    -6     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     1    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     1    -6     0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     1     0     0    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     1     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     0     0     1     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0     1     0     0    -6     1     0     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0    -6     1     0     1     0     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     1    -6     1     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     1    -6     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     0    -6     1     0     1     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     1     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1     0     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     0    -6     1     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     1    -6
```

2.46 Generate the adjugate matrix for square matrix

Given square matrix such as

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

find the adjugate matrix which is

$$\begin{pmatrix} -3 & 6 & -3 \\ 6 & -12 & 6 \\ -3 & 6 & -3 \end{pmatrix}$$

Mathematica

```
<<"Combinatorica`" (*not needed in V9*)
mat      = {{1,2,3},{4,5,6},{7,8,9}};
cof      = Table[Cofactor[mat,{i,j}],
              {i,3},{j,3}];
adjugate = Transpose[cof]
```

```
{ {-3,   6, -3},
 { 6, -12,  6},
 {-3,   6, -3}}
```

Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
LinearAlgebra:-Adjoint(A)
```

$$\begin{bmatrix} -3 & 6 & -3 \\ 6 & -12 & 6 \\ -3 & 6 & -3 \end{bmatrix}$$

Matlab Will try to find function in Matlab to do this. But for non-singular matrices only the direct method of finding the inverse and then multiplying by the determinant to recover the adjunct can be used.

```
A=[1 2 3;4 5 6;7 8 9]
det(A)*inv(A)
```

```
Warning: Matrix is close to singular
or badly scaled. Results may be inaccurate.
RCOND = 1.541976e-18.

ans =
   -3.0000    6.0000   -3.0000
    6.0000  -12.0000    6.0000
   -3.0000    6.0000   -3.0000
```

The following is due to Matt J from the matlab newsgroup

```
A=[1 2 3;
   4 5 6;
   7 8 9]

c=poly(A);
adjunct_A=polyvalm(c(1:end-1),A)
```

```
adjunct_A =

 -3.0000    6.0000   -3.0000
  6.0000  -12.0000    6.0000
 -3.0000    6.0000   -3.0000
```

Fortran

Thanks goes to James Van Buskirk and Louisa from comp.lang.fortran for the review and suggestions which lead to improvements of the code.

```
 !-- Find the Adjugate of a square matrix
!-- Version 6/22/2012 by Nasser M. Abbasi gfortran 4.6.3
program t44
implicit none
integer, parameter :: n=3
integer :: INFO,i,j,ii,IPIV(n-1),number_row_exchange
real (kind=kind(0.0d0)) :: A(n,n),B(n-1,n-1),C(n,n)
logical :: M(n,n)

A(1,:) = [1, 2, 3];
A(2,:) = [4, 5, 6];
A(3,:) = [7, 8, 9];

DO j=1,n
```

```
   DO i=1,n

      M = .true.
      M(:,j) = .false.
      M(i,:) = .false.
      B = reshape(pack(A, M),[n-1,n-1])

      !-- LU decomposition in order to obtain determinant
      CALL DGETRF( n-1, n-1, B, n-1, IPIV, INFO )

      !-- determinant of U is the product of diagonal
      C(i,j)= PRODUCT( [(B(ii,ii),ii=1,n-1)] )

      !-- Adjust sign based on number of row exchanges and (-1)^(i+j)
      number_row_exchange = count(IPIV /= [(ii,ii=1,n-1)])
      C(i,j) = C(i,j)*(1-2*MODULO(number_row_exchange+i+j,2))

   END DO
END DO
write(*,'(3F15.4)') C
end program t44
```

```
#compile, link and run
export LD_LIBRARY_PATH=/usr/lib/atlas-base/:/usr/lib/atlas-base/atlas/
gfortran -fcheck=all -Wall t44.f90 -L/usr/lib/atlas-base/atlas/ -lblas -llapack
>./a.out
        -3.0000         6.0000        -3.0000
         6.0000       -12.0000         6.0000
        -3.0000         6.0000        -3.0000
!>dpkg -l|grep -E "(blas|lapack)"
!ii  libblas3gf        1.2.20110419-2ubuntu1  Basic Linear Algebra Reference implementations
!ii  liblapack-dev     3.3.1-1                library of linear algebra routines 3 - static
!ii  liblapack-doc     3.3.1-1                library of linear algebra routines 3 - documen
!ii  liblapack3gf      3.3.1-1                library of linear algebra routines 3 - shared
!ii  libopenblas-base  0.1alpha2.2-3          Optimized BLAS (linear algebra) library based
!ii  libopenblas-dev   0.1alpha2.2-3          Optimized BLAS (linear algebra) library based
!>
```

Ada

```
-- Ada implementation
with Ada.Text_Io; use Ada.Text_Io;
with Ada.Float_Text_Io; use Ada.Float_Text_Io;
with Ada.integer_Text_Io; use Ada.integer_Text_Io;
with Ada.Numerics.Real_Arrays;  use Ada.Numerics.Real_Arrays;
procedure t44 is
    A : constant real_matrix :=
             (( 1.0,  2.0,  3.0),
              ( 4.0,  5.0,  6.0),
              ( 7.0,  8.0,  9.0));
           --(( -3.0,  2.0,  -5.0),
           --   ( -1.0,  0.0,  -2.0),
           --   ( 3.0,  -4.0,  1.0));
    C : real_matrix(A'range(1), A'range(2));
    B : real_matrix(1 .. 2, 1 .. 2);
    -- Thanks goes to Dmitry A. Kazakov this function
    function Exclude (A : Real_Matrix; I, J : Integer)
       return Real_Matrix is
       AI, AJ : Integer := A'First (1);
    begin
       return B : Real_Matrix (1..A'Length (1)-1, 1..A'Length (2)-1) do
          AI := A'First (1);
          for BI in B'Range (1) loop
             if AI = I then
                AI := AI + 1;
             end if;
             AJ := A'First (2);
             for BJ in B'Range (2) loop
                if AJ = J then
                   AJ := AJ + 1;
                end if;
                B (BI, BJ) := A (AI, AJ);
                AJ := AJ + 1;
             end loop;
             AI := AI + 1;
          end loop;
       end return;
    end Exclude;
    -- Function to print a 2D matrix
```

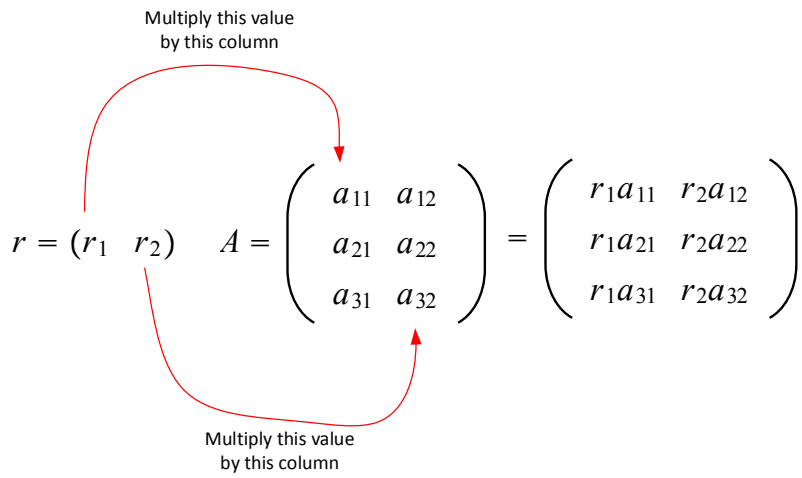
```
    procedure Put (X : Real_Matrix) is
       begin
          for I in X'Range (1) loop
             for J in X'Range (2) loop
                Put (X (I, J) ); put("    ");
             end loop;
             New_Line;
          end loop;
   end Put;
begin
    FOR I in A'range(1) LOOP
      FOR J in A'range(2) LOOP
          B := Exclude (A, I, J);
          C(I,J) := float((-1)**( ((I+J) rem 2) ))*determinant(B);
      END LOOP;
    END LOOP;
    c := transpose(c);
    put(c);
end t44;
```

compile and run

```
-- export LD_LIBRARY_PATH=/usr/lib/atlas-base/:/usr/lib/atlas-base/atlas/
>gnatmake t44.adb
gcc-4.6 -c t44.adb
gnatbind -x t44.ali
gnatlink t44.ali

>./t44
-3.00000E+00     6.00000E+00    -3.00000E+00
 6.00000E+00    -1.20000E+01     6.00000E+00
-3.00000E+00     6.00000E+00    -3.00000E+00
```

2.47 Multiply each column by values taken from a row

Given a row of the same number of values as the number of columns in a matrix, how to scale each column by the corresponding value in that row? This picture explains more the problem



In Matlab, bsxfun is used.

2.47.1 Mathematica

credit for this solution goes to Bob Hanlon, Adriano Pascoletti, Kurt Tekolste, David Park, and Peter. J. C. Moses from the Math group

```
r   = {2,3};
mat = {{1,2},{3,4},{5,6}};
r #&/@mat

(*or*)
Map[v*#&, mat]
```

```
   {{2,  6},
    {6,  12},
    {10, 18}}
```

Another way is to use Inner[] command. Credit for this solution goes to Sswziwa Mukasa and Peter. J. C. Moses from the Math group

```
r={2,3};
mat={{1,2},{3,4},{5,6}};
Inner[Times,mat,r,List]
```

```
Out[66]= {{2,  6},
          {6,  12},
          {10, 18}}
```

2.47.2 Matlab

```
r=[2 3];
A=[1 2;
   3 4;
   5 6]
bsxfun(@times,r,A)
```

```
ans =
     2     6
     6    12
    10    18
```

2.47.3 Fortran

```
program t45
implicit none

integer, parameter :: nRow=3, nCol=2

character(len=*), parameter :: FMT = "(2I5)"
integer :: v(nCol),A(nRow,nCol),B(nRow,nCol),i,j;

!-- initialization of data
v = [2,3]
A(1,:) = [1,2];
A(2,:) = [3,4];
A(3,:) = [5,6];
```

```
 !F2008
 !do concurrent (j = 1:nCol)
 !   B(:,j) = v(j)*A(:,j)
 !end do

 do j=1,nCol
    B(:,j) = v(j)*A(:,j)
 end do

 write(*,FMT) ( (B(i,j),j=1,nCol), i=1,nRow)

end program t45
```

```
#compile and run
>gfortran -std=f2008 t45.f90
>./a.out
    2    6
    6   12
   10   18
```

2.47.4 Octave

Use automatic broadcasting

```
r .* A
```

2.47.5 Maple

```
r:=Vector[row]([2,3]);
mat:=Matrix([[1,2],[3,4],[5,6]]);

f:=n->r(n)*mat(1..3,n);
f~([1,2]);
convert(%,Matrix)
```

$$\begin{bmatrix} 2 & 6 \\ 6 & 12 \\ 10 & 18 \end{bmatrix}$$

2.48 extract submatrix from a larger matrix by removing row/column

Given 2D matrix A extract all submatrices by removing row/column.

For example, given the matrix $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ then the submatrix for $(1,1)$ is $\begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix}$ obtained by removing the first row and the first column.

In Mathematica, ReplacePart can be used. In Matlab the [] operator can be used. In Fortran, the pack() function can be used.

2.48.1 Mathematica

```
mat={{1,2,3},
     {4,5,6},
     {7,8,9}};
{nRow, nCol} = Dimensions[mat];
Table[ReplacePart[mat,{{i},{i,_},{_,j}}
      :>Sequence[],Heads-> False],
         {i,nRow},{j,nCol}];

r=Flatten[%,1]
```

```
{{{5,6},{8,9}},
 {{4,6},{7,9}},
 {{4,5},{7,8}},
 {{2,3},{8,9}},
 {{1,3},{7,9}},
 {{1,2},{7,8}},
 {{2,3},{5,6}},
 {{1,3},{4,6}},
 {{1,2},{4,5}}}
```

2.48.2 Matlab

```
A=[1 2 3;
   4 5 6;
   7 8 9]

for i=1:size(A,1)
    for j=1:size(A,1)
        B=A;
        B(i,:)=[];
        B(:,j)=[]
    end
end
```

```
B =
     5     6
     8     9
B =
     4     6
     7     9
B =
     4     5
     7     8
B =
     2     3
     8     9
B =
     1     3
     7     9
B =
     1     2
     7     8
B =
     2     3
     5     6
B =
     1     3
     4     6
B =
     1     2
     4     5
```

2.48.3 Fortran

```
 program t46
implicit none
integer, parameter:: dp=kind(0.d0), n=3
integer :: i,j,ii
real(dp) :: A(n,n) = dble(reshape([ 1.0, 2.0, 3.0,  &
                                    4.0, 5.0, 6.0,  &
                                    7.0, 8.0, 9.0], &
                                    shape(A),       &
                                    order=[2,1]))
real(dp) :: B(n-1,n-1)
logical :: mask(n,n)
DO i=1,n
   DO j=1,n
      mask = .true.
      mask(:,j) = .false.
      mask(i,:) = .false.
      !-- extract submatrix
      B = reshape(pack(A, mask),[n-1,n-1])

      DO ii=1,n-1
         write(*,'(2F6.1)') B(ii,:)
      END DO
      write(*,'(/)')
   END DO
END DO
end program t46
```

compile and run

```
>gfortran -std=f2008 -fcheck=all -Wall -Wextra -Wconversion-extra t46.f90
>./a.out
   5.0   6.0
   8.0   9.0

   4.0   6.0
   7.0   9.0

   4.0   5.0
   7.0   8.0
```

```
  2.0   3.0
  8.0   9.0

  1.0   3.0
  7.0   9.0

  1.0   2.0
  7.0   8.0

  2.0   3.0
  5.0   6.0

  1.0   3.0
  4.0   6.0

  1.0   2.0
  4.0   5.0
```

2.48.4 Ada

```
with Ada.Text_Io; use Ada.Text_Io;
with Ada.Float_Text_Io; use Ada.Float_Text_Io;
with Ada.Numerics.Real_Arrays; use Ada.Numerics.Real_Arrays;
procedure foo is
    A : constant Real_Matrix :=
             (( 1.0,  2.0,  3.0),
              ( 4.0,  5.0,  6.0),
              ( 7.0,  8.0,  9.0));
    B : real_matrix(1 .. 2, 1 .. 2);
    -- Thanks goes to Dmitry A. Kazakov
    -- for providing this function
    function Exclude (A : Real_Matrix; I, J : Integer)
       return Real_Matrix is
       AI, AJ : Integer := A'First (1);
    begin
       return B : Real_Matrix(1..A'Length(1)-1,1..A'Length(2)-1) do
          AI := A'First (1);
          for BI in B'Range (1) loop
             if AI = I then
```

```
                AI := AI + 1;
             end if;
             AJ := A'First (2);
             for BJ in B'Range (2) loop
                if AJ = J then
                   AJ := AJ + 1;
                end if;
                B (BI, BJ) := A (AI, AJ);
                AJ := AJ + 1;
             end loop;
             AI := AI + 1;
          end loop;
       end return;
    end Exclude;

    -- Function to print a 2D matrix
    procedure put(A: Real_Matrix) is
    begin
     for I in A'range(1) loop
      for J in A'range(2) loop
          put(A(I,J));
      end loop;
      new_line;
     end loop;
    end put;

begin
    FOR I in A'range(1) LOOP
      FOR J in A'range(2) LOOP
          B := Exclude (A, I, J);
          put(B);
          new_line(1);
      END LOOP;
    END LOOP;
end foo;
```

compile and run

```
>gnatmake foo.adb
gcc-4.6 -c foo.adb
gnatbind -x foo.ali
```

```
gnatlink foo.ali
>./foo
 >./foo
  5.00000E+00 6.00000E+00
  8.00000E+00 9.00000E+00

  4.00000E+00 6.00000E+00
  7.00000E+00 9.00000E+00

  4.00000E+00 5.00000E+00
  7.00000E+00 8.00000E+00

  2.00000E+00 3.00000E+00
  8.00000E+00 9.00000E+00

  1.00000E+00 3.00000E+00
  7.00000E+00 9.00000E+00

  1.00000E+00 2.00000E+00
  7.00000E+00 8.00000E+00

  2.00000E+00 3.00000E+00
  5.00000E+00 6.00000E+00

  1.00000E+00 3.00000E+00
  4.00000E+00 6.00000E+00

  1.00000E+00 2.00000E+00
  4.00000E+00 5.00000E+00
```

2.48.5 Maple

```
del:=proc(A::Matrix,idx::list)::Matrix;
     local B;
     B:=LinearAlgebra:-DeleteRow(A,idx[1]);
     B:=LinearAlgebra:-DeleteColumn(B,idx[2]);
     return B;
end proc;

A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]):
```

```
r:=map2(del,A,[indices(A,indexorder)]);
```

$$\left[\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}, \begin{bmatrix} 4 & 6 \\ 7 & 9 \end{bmatrix}, \begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}, \begin{bmatrix} 2 & 3 \\ 8 & 9 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 7 & 9 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 7 & 8 \end{bmatrix}, \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}, \begin{bmatrix} 1 & 3 \\ 4 & 6 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}\right]$$

2.49 delete one row from a matrix

Example, Given the folowing 2D matrix A delete the second row

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

2.49.1 Mathematica

```
mat={{1,2,3},
     {4,5,6},
     {7,8,9}};

mat[[2]]=Sequence[];
mat
```

```
Out[81]= {{1,2,3},
          {7,8,9}}
```

or a little longer solution using Pick

```
{nRow,nCol}=Dimensions[mat];
mask=Table[True,{nRow},{nCol}];
mask[[2]]=False;
Pick[mat,mask]
```

```
Out[70]= {{1,2,3},
          {7,8,9}}
```

2.49.2 Matlab

```
A=[1 2 3;4 5 6;7 8 9];
A(2,:)=[]
```

```
A =
     1     2     3
     7     8     9
```

2.49.3 Maple

```
A:=<1,2,3;
    4,5,6;
    7,8,9>;
A:=LinearAlgebra:-DeleteRow(A,2);
```

```
     [1  2  3]
A := [       ]
     [7  8  9]
```

2.50 delete one column from a matrix

Example, Given the folowing 2D matrix A delete say the second column

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

2.50.1 Mathematica

```
mat={{1,2,3},
     {4,5,6},
     {7,8,9}};

mat[[All,2]]=Sequence[];
mat
```

```
Out[93]= {{1,3},
          {4,6},
          {7,9}}
```

or a little longer solution using Pick

```
{nRow,nCol}=Dimensions[mat];
mask=Table[True,{nRow},{nCol}];
mask[[All,2]]=False;
mat=Pick[mat,mask];
mat
```

```
Out[98]= {{1,3},
          {4,6},
          {7,9}}
```

2.50.2 Matlab

```
A=[1 2 3;4 5 6;7 8 9];
A(:,2)=[]
```

```
A =
     1     3
     4     6
     7     9
```

2.50.3 Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
A:=LinearAlgebra:-DeleteColumn(A,2);
```

$$\begin{bmatrix} 1 & 3 \\ 4 & 6 \\ 7 & 9 \end{bmatrix}$$

2.51 generate random matrix so that each row adds to 1

Generate the random matrix and divide each row by its total

2.51.1 Mathematica

```
mat = Table[RandomReal[],{3},{4}]
```

$$\begin{pmatrix} 0.76862 & 0.917299 & 0.606119 & 0.63735 \\ 0.610077 & 0.208307 & 0.0337861 & 0.473017 \\ 0.388772 & 0.432688 & 0.475881 & 0.68523 \end{pmatrix}$$

```
s   = Total[mat,{2}]
```

```
{1.15201, 1.99068, 3.05063}
```

```
s   = Total[mat,{2}];
b   = mat/s
```

$$\begin{pmatrix} 0.15919 & 0.0553157 & 0.448352 & 0.337142 \\ 0.358594 & 0.264968 & 0.123659 & 0.25278 \\ 0.216269 & 0.248665 & 0.278871 & 0.256195 \end{pmatrix}$$

```
Total[b,{2}]
```

```
Out[24]= {1.,1.,1.}
```

2.51.2 Matlab

```
A = rand(3,4)
s = sum(A,2);
B = bsxfun(@rdivide,A,s)
sum(B,2)
```

```
A =
 0.6787    0.3922    0.7060    0.0462
 0.7577    0.6555    0.0318    0.0971
 0.7431    0.1712    0.2769    0.8235

B =
 0.3723    0.2151    0.3873    0.0253
 0.4913    0.4250    0.0206    0.0630
 0.3689    0.0850    0.1375    0.4087

ans =
    1.0000
    1.0000
    1.0000
```

2.51.3 Maple

```
restart;
A:=LinearAlgebra:-RandomMatrix(3,4,generat
for n from 1 to LinearAlgebra:-RowDimensio
   A[n,..]:=A[n,..]/add(A[n,..]);
od:
A;

#verify each row sums to 1
map(x->add(x),[LinearAlgebra:-Row(A,[1..-1
```

```
      [0.9705927818  0.9575068354  0.0975404050
      [
 A := [0.1576130817  0.5468815192  0.6323592462
      [
      [0.9648885352  0.2784982189  0.9133758561

    [0.4508876154  0.4448085560  0.0453122684  0.
    [
    [0.0702799715  0.2438555046  0.2819701848  0.
    [
    [0.3247157951  0.0937235414  0.3073801341  0.


     [1.00000000026568, 1.00000000004739, 1.00000
```

2.52 generate random matrix so that each column adds to 1

Generate the random matrix and divide each column by its total

2.52.1 Mathematica

This method due to Bob Hanlon from the Math group

```
mat =  Table[RandomReal[],{3},{4}]
s   = Total[mat,{1}]
b   = #/s& /@ mat
Total[b,{1}]
```

```
Out[31]= {{0.440393,0.945076,0.0527301,0.537288},
          {0.515868,0.565691,0.800959,0.0302484},
          {0.509004,0.143124,0.519455,0.264779}}

Out[32]= {1.46526,1.65389,1.37314,0.832315}

Out[36]= {{0.300555,0.571426,0.038401,0.645535},
          {0.352065,0.342036,0.583303,0.0363425},
          {0.34738,0.0865376,0.378296,0.318123}}

Out[37]= {1.,1.,1.,1.}
```

Or can use Transpose

```
b   = Transpose[Transpose[mat]/s];
Total[b,{1}]
```

```
{1.,1.,1.,1.}
```

Another way of doing the above, without the need to transpose 2 times is the following

```
b=Inner[Divide,mat,s,List];
Total[b,{1}]
```

```
{1.,1.,1.,1.}
```

2.52.2 Matlab

```
A=rand(3,4)
s=sum(A,1);
B=bsxfun(@rdivide,A,s)
sum(B,1)
```

```
A =
    0.6948    0.0344    0.7655    0.4898
    0.3171    0.4387    0.7952    0.4456
    0.9502    0.3816    0.1869    0.6463

B =
    0.3541    0.0403    0.4380    0.3097
    0.1616    0.5133    0.4550    0.2817
    0.4843    0.4464    0.1069    0.4086

ans =
    1.0000    1.0000    1.0000    1.0000
```

2.52.3 Maple

```
A:=LinearAlgebra:-RandomMatrix(3,4);
```

```
     [ 9  -95   51  24]
     [                ]
A := [99  -20   76  65]
     [                ]
     [60  -25  -44  86]
```

```
b:=MTM:-sum(A,1);  #sum columns
     b := [168, -140, 83, 175]
```

```
nCol:=LinearAlgebra:-ColumnDimension(A):
B:=convert([seq(A[..,i]/b[i],i=1..nCol)],Matrix);
```

```
     [3   19   51  24 ]
     [--  --   --  ---]
     [56  28   83  175]
     [                ]
     [33   1   76   13]
B := [--   -   --   --]
     [56   7   83   35]
     [                ]
```

```
[5   5   -44  86 ]
[--  --  ---  ---]
[14  28  83   175]
```

```
MTM:-sum(B,1);
```

```
  [1, 1, 1, 1]
```

2.53 sum all rows in a matrix

Given the folowing 2D matrix A find the sum of each row

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

2.53.1 Mathematica

```
mat={{1,2,3},
      {4,5,6},
      {7,8,9}}
Total[mat,{2}]
```

```
Out[2]= {6,
         15,
         24}
```

2.53.2 Matlab

```
A=[1 2 3;4 5 6;7 8 9];
sum(A,2)
```

```
ans =
     6
    15
    24
```

2.53.3 Maple

```
A:=Matrix([[1,2,3],[4,5,6],[7,8,9]]);
map(x->add(x),convert(A,listlist))

#or may be better is
`add`~([LinearAlgebra:-Row(A,[1..-1])])
```

```
[6, 15, 24]
```

2.54 sum all columns in a matrix

Given the folowing 2D matrix A find the sum of each column

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Mathematica

```
In[3]:= mat={{1,2,3},
             {4,5,6},
             {7,8,9}}
Total[mat,{1}]

Out[4]= {12,15,18}
```

Matlab

```
A=[1 2 3;4 5 6;7 8 9];
sum(A,1)

ans =
    12    15    18
```

Maple

```
`add`~([LinearAlgebra:-Column(A,[1..-1])])

#      [12, 15, 18]
```

2.55 find in which columns values that are not zero

given matrix $\begin{pmatrix} 0 & 39 & 0 \\ 55 & 100 & 0 \\ 34 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 50 \end{pmatrix}$ find the column index which contain values > 0 but do it from top row to the bottom row. Hence the result should be $\begin{pmatrix} \overbrace{2}^{1st\ row}, \overbrace{1,2}^{2nd\ row}, \overbrace{1}, \overbrace{2}, \overbrace{3} \end{pmatrix}$ this is becuase on the first row, nonzero is in second column, and on the second row, nonzero is at first and second column, and on the third row, nonzero is at the first column, and so on.

Mathematica

```
mat={{0, 39,  0},
     {55,100, 0},
     {34,0,   0},
     {0, 9,   0},
     {0, 0,   50}};
Position[#,x_/;x>0]&/@mat;
Flatten[%]
```

```
Out[18]= {2,1,2,1,2,3}
```

Matlab

```
A=[0  39   0
   55 100  0
   34 0    0
   0  9    0
   0  0   50];
[I,~]=find(A'>0)
I'
```

```
2     1     2     1     2     3
```

Maple

```
A:=Matrix([[0,39,0],
           [55,100,0],
           [34,0,0],
           [0,9,0],
           [0,0,50]]):
f:=x->`if`(not evalb(A(x[1],x[2])=0),x[2],NULL):
f~([indices(A,indexorder)])

#Or may be better is
map(x->convert(ArrayTools:-SearchArray(x),list),[LinearAlgebra:-Row(A,[1..-1])]);
ListTools:-Flatten(%)
```

```
[2,1,2,1,2,3]
```

2.56 How to remove values from one vector that exist in another vector

Given vector $v1 = \{1, 3, 4, 5, 8, 9, 20.2, 30, -44\}$ remove from it any element that exist in $v2 = \{1, 7, 4\}$. Notice that `Complement[]` in Mathematica or `setdiff()` in Matlab can be used, but they will sort the result. Hence they are not used here in order to keep the order the same.

2.56.1 Mathematica

```
#first method
v1={1,3,4,5,8,9,20.2,30,-44};
v2={1,7,4};
v1=DeleteCases[v1,Alternatives@@v2]
```

{3,5,8,9,20.2,30,-44}

```
#second method
v1={1,3,4,5,8,9,20.2,30,-44};
v2={1,7,4};
v1=DeleteCases[v1,x_/;MemberQ[v2,x]]
```

{3,5,8,9,20.2,30,-44}

2.56.2 Matlab

```
v1=[1,3,4,5,8,9,20.2,30,-44];
v2=[1,7,4];
v1(ismember(v1,v2))=[]
```

```
v1 =
3.0000 5.0000 8.0000 9.0000 20.2000 30.0000 -44.0000
```

2.56.3 Fortran

```
 program f
  implicit none
  REAL, ALLOCATABLE :: A(:),B(:)
  A =[1.0,3.0,4.0,5.0,8.0,9.0,20.2,30.0,-44.0];
  B =[1.0,7.0,4.0];
  A = pack(A, A .NE. B)
  Print *,A
end program f
```

```
>gfortran f2.f90
>./a.out
3.0000000 5.0000000 8.0000000  9.0000000  20.200001  30.000000 -44.000000
```

2.56.4 Maple

```
v1:=Array([1,3,4,5,8,9,20.2,30,-44]);
v2:=Array([1,7,4]);
select[flatten](x->not(member(x,v2)),v1)
```

```
[3, 5, 8, 9, 20.2, 30, -44]
```

2.57 How to find mean of equal sized segments of a vector

Given vector V of length m find the mean of segments $V(1:n), V(n+1:2n), V(2n+1:3n)\ldots$. In otherwords, equal length segments.

Mathematica

```
len = 52; n = 4;
v   = RandomReal[{0, 1}, len];
Mean /@ Partition[v, n]
```

```
{0.750653,0.410073,0.401005,
0.138907,0.466247,0.568257,
0.535362,0.517755,0.555368,
0.705857,0.502319,0.453571,0.357949}
```

Matlab

```
N = 52;
n = 4;
V = rand(N,1);
mean(reshape(V,n,N/n))'
```

```
ans =
    0.3026
    0.3589
    0.4745
    0.6249
    0.3042
    0.5428
    0.2387
    0.3200
    0.6224
    0.4408
    0.5657
    0.5469
    0.5696
```

2.58 find first value in column larger than some value and cut matrix from there

Given matrix

$$\begin{pmatrix} 1 & 5 \\ 2 & 3 \\ 4 & 8 \\ 7 & 2 \end{pmatrix}$$

search in column 2 of matrix for the first time a value exceeds 6 and return the matrix up to that row. The result should be

$$\begin{pmatrix} 1 & 5 \\ 2 & 3 \end{pmatrix}$$

Mathematica

```
a = {{1, 5}, {2, 3}, {4, 8}, {7, 2}};
Min@Position[a, {x_, y_} /; y > 6]
(* 3 *)

a[[1 ;; % - 1]]
```

```
{{1, 5}, {2, 3}}
```

Matlab

```
A = [1,5;2,3;4,8;7,2];
A(1:find(A(:,2)>6)-1,:)
```

```
ans =
     1     5
     2     3
```

Maple

```
A:=Matrix([[1,5],[2,3],[4,8],[7,2]]);
idx := ListTools:-SelectFirst(1, x->x>6 , A[..,2],output=indices);
if idx[1]>1 then
   A[1..idx[1]-1,..];
else
   A[1..idx[1],..];
fi;
```

$$\begin{bmatrix} 1 & 5 \\ 2 & 3 \end{bmatrix}$$

2.59 make copies of each value into matrix into a larger matrix

Do the following transformation. Take each element of matrix, and replace it by a 2 by 2 matrix with its values in places.

kron() in Matlab and KroneckerProduct in Mathematica and Maple can be used for this.

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 2 & 2 & 3 & 3 \\ 1 & 1 & 2 & 2 & 3 & 3 \\ 4 & 4 & 5 & 5 & 6 & 6 \\ 4 & 4 & 5 & 5 & 6 & 6 \\ 7 & 7 & 8 & 8 & 9 & 9 \\ 7 & 7 & 8 & 8 & 9 & 9 \end{pmatrix}$$

Mathematica

```
mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
kernel = {{1, 1}, {1, 1}};
KroneckerProduct[mat, kernel]
```

```
{{1, 1, 2, 2, 3, 3},
 {1, 1, 2, 2, 3, 3},
 {4, 4, 5, 5, 6, 6},
 {4, 4, 5, 5, 6, 6},
 {7, 7, 8, 8, 9, 9},
 {7, 7, 8, 8, 9, 9}}
```

Another method that can be used in this, but I prefer the kron method above:

```
mat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
Map[ConstantArray[#, {2, 2}] &, mat, {2}]
ArrayFlatten[%, 2]
```

Matlab

```
A=[1 2 3; 4 5 6; 7 8 9]
kernel=ones(2,2)
kron(A,kernel)
```

```
1   1   2   2   3   3
1   1   2   2   3   3
4   4   5   5   6   6
4   4   5   5   6   6
7   7   8   8   9   9
7   7   8   8   9   9
```

Maple

```
restart;
mat := Matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);
kernel :=  Matrix([[1, 1], [1, 1]]);
LinearAlgebra:-KroneckerProduct(mat,kernel)
```

$$\begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 \\ 1 & 1 & 2 & 2 & 3 & 3 \\ 4 & 4 & 5 & 5 & 6 & 6 \\ 4 & 4 & 5 & 5 & 6 & 6 \\ 7 & 7 & 8 & 8 & 9 & 9 \\ 7 & 7 & 8 & 8 & 9 & 9 \end{bmatrix}$$

2.60 repeat each column of matrix number of times

Gives a matrix, repeate each column a number of time, say 3 times, in place to produce a matrix 3 times as wide as the original.

kron() in Matlab and KroneckerProduct in Mathematica and Maple can be used for this.

Mathematica

```
mat = {{1, 2, 3, 4},
       {5 , 6, 7, 8},
       {9, 10, 11, 12}}

KroneckerProduct[mat, {1, 1, 1}]
```

```
{{1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4},
 {5, 5, 5, 6, 6, 6, 7, 7, 7, 8, 8, 8},
 {9, 9, 9, 10, 10, 10, 11, 11, 11, 12, 12, 12}}
```

Another method that can be used in this, but the above is better

```
r = Map[#*{1, 1, 1} &, mat, {2}]
Partition[Flatten[r], 12]
```

Matlab

```
A=[1 2 3 4;
   5 6 7 8;
   9 10 11 12];
kernel=ones(1,3);
kron(A,kernel)
```

```
ans =
1  1  1  2  2   2  3   3  3  4  4   4
5  5  5  6  6   6  7   7  7  8  8   8
9  9  9  10 10  10 11  11 11 12 12  12
```

Maple

```
restart;
mat := Matrix([[1, 2, 3,4], [5, 6,7,8], [9,10,11,12]]);
kernel :=  Vector[row]([1,1,1]);
LinearAlgebra:-KroneckerProduct(mat,kernel);
```

$$\begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 5 & 5 & 5 & 6 & 6 & 6 & 7 & 7 & 7 & 8 & 8 & 8 \\ 9 & 9 & 9 & 10 & 10 & 10 & 11 & 11 & 11 & 12 & 12 & 12 \end{bmatrix}$$

2.61 How to apply a function to each value in a matrix?

Apply function $f(x,n)$ to ragged matrix where n is value taken from the matrix and x is some other value defined somewhere else. The result of the function should replace the same position in the matrix where n was.

For example, given matrix

$$mat = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & \\ 7 & 8 & 9 \end{pmatrix}$$

generate the matrix

$$\begin{pmatrix} f[x,1] & f[x,2] & f[x,3] \\ f[x,4] & f[x,5] & \\ f[x,7] & f[x,8] & f[x,9] \end{pmatrix}$$

Mathematica

The trick is to use `Map` with `2` at end.

```
ClearAll[f, x]
mat = {{1, 2, 3}, {4, 5}, {7, 8, 9}};
Map[f[x, #] &, mat, {2}]
```

$$\{\{f(x,1),f(x,2),f(x,3)\},\{f(x,4),f(x,5)\},\{f(x,7),f($$

Maple

```
#use double map
restart;
mat:=[[1,2,3],[4,5],[7,8,9]];
map(x->map(y->f(y),x),mat)
```

```
[[f(1), f(2), f(3)], [f(4), f(5)], [f(7), f(8), f
```

2.62 How to sum all numbers in a list (vector)?

Given a vector or list $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ how to find the sum of all its elements?

Mathematica

```
list=Range[10]
```

```
{1,2,3,4,5,6,7,8,9,10}
```

```
Total[list]
```

55

Matlab

```
lst=1:10
sum(lst)
```

55

Maple

```
lst:=[seq( i, i=1..10 )];
add(lst)
```

55

2.63 How to find maximum of each row of a matrix?

Given

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Find the maximum of each row. The result should be $(2,4)$. (to find the min, just change Max with Min below.

Mathematica

```
mat = {{1, 2}, {3, 4}}
Map[Max , mat]  (* or  Max /@ mat  *)
```

```
{2,4}
```

Matlab

```
A=[1 2;3 4];
max(A')
```

```
     2     4
```

Maple

```
A:=Matrix([[1,2],[3,4]]);
`max`~([LinearAlgebra:-Row(A,1..-1)])
```

```
[2, 4]
```

2.64 How to find maximum of each column of a matrix?

Given

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Find the maximum of each column. The result should be $(3,4)$. (to find the min, just change Max with Min below.

Mathematica

```
mat = {{1, 2}, {3, 4}}
Map[Max , Transpose[mat]]

(* or Max /@ Transpose@mat *)
```

```
{3, 4}
```

Matlab

```
A=[1 2;3 4];
max(A)
```

```
     3     4
```

Maple

```
A:=Matrix([[1,2],[3,4]]);
`max`~([LinearAlgebra:-Column(A,1..-1)])
```

```
[3, 4]
```

2.65 How to add the mean of each column of a matrix from each column?

Given

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Find the mean of each column, and add this mean to each element of the corresponding column. The result should be

$$\begin{pmatrix} 3 & 5 \\ 5 & 7 \end{pmatrix}$$

To subtract the mean, just replace Plus with Subtract below for Mathematica and replace `@plus` with `@minus` for Matlab. This shows that Matlab `bsxfun` is analogue to Mathematica's `MapThread`.

Mathematica

```
mat = {{1, 2}, {3, 4}}
MapThread[Plus,{Transpose@mat, Mean[mat]}]
```

```
{{3, 5}, {5, 7}}
```

Matlab

```
A=[1 2;3 4];
bsxfun(@plus, A, mean(A))
```

```
ans =
     3     5
     5     7
```

Maple

Maple mean always return real value, even if exact is possible.

```
A:=Matrix([[1,2],[3,4]]);
for n from 1 to LinearAlgebra:-ColumnDimension(A) do
    A[..,n]:=A[..,n] +~ Statistics:-Mean(A[..,n])
od;
A
```

$$\begin{bmatrix} 3.0 & 5.0 \\ 5.0 & 7.0 \end{bmatrix}$$

2.66 How to add the mean of each row of a matrix from each row?

Given

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Find the mean of each row, and add this mean to each element of the corresponding row. The result should be

$$\begin{pmatrix} 2.5 & 3.5 \\ 6.5 & 7.5 \end{pmatrix}$$

To subtract the mean, just replace Plus with Subtract below for Mathematica and replace `@plus` with `@minus` for Matlab. This shows that Matlab `bsxfun` is analogue

to Mathematica's `MapThread`.. The main difference is that Matlab is a little bit more consistent in this example, since in Matlab all operations are done column wise. Hence `mean(A)` takes the mean of each column (same as Matematica in this one case), but `bsxfun` also acts column wise on the matrix `A`, while Mathematica Map and MapThread act row wise (list of lists). One just needs to be careful about this order difference.

Mathematica

```
mat = {{1, 2}, {3, 4}};
MapThread[Plus,{mat, Mean@Transpose@mat}] // N
```

```
{{2.5, 3.5}, {6.5, 7.5}}
```

Matlab

```
A=[1 2;3 4];
bsxfun(@plus, A', mean(A'))
```

```
ans =
          2.5          6.5
          3.5          7.5
```

Maple

```
A:=Matrix([[1,2],[3,4]]);
for n from 1 to LinearAlgebra:-RowDimension(A) do
    A[n,..]:=A[n,..] +~ Statistics:-Mean(A[n,..])
od;
A
```

$$\begin{bmatrix} 2.5000000000 & 3.5000000000 \\ 6.5000000000 & 7.5000000000 \end{bmatrix}$$

2.67 Find the different norms of a vector

Problem: Given the vector say

$$v = 1, 2, 4$$

Find its norm for $p = 1, 2, \infty$

Mathematica

```
Remove["Global`*"];

v = {{1}, {2}, {4}};
p = {1, 2, Infinity};
Map[Norm[v,#]&,p]
```

```
{7,Sqrt[21],4}
```

Matlab

```
clear all; close all;
v=[1 2 4];
p=[1,2,inf];
arrayfun(@(i) norm(v,p(i)),1:length(p))
```

```
ans =
7.0000    4.5826    4.0000
```

2.68 Check if a matrix is Hermite

Problem: Given a matrix A, check to see if it is Hermite.

A Matrix is Hermite if it is the same as the conjugate of its transpose. One way to check is to take the difference and check that all values in the resulting difference matrix are zero.

To account for numerical values, the check is done using machine epsilon.

Mathematica

```
mat = {{-1,1-3*I,0},
       {1+3*I,0,-6*I},
       {0,6*I,1}};
HermitianMatrixQ[mat]
```

True

Matlab

```
clear all;
i=sqrt(-1);
mat=[-1      1-3*i     0;
    1+3*i     0       -6*i;
      0       6*i      1];

mat2=conj(transpose(mat));
diff = mat-mat2
```

```
diff =
     0     0     0
     0     0     0
     0     0     0
```

```
r=abs(diff)<eps('double')
```

```
r =
     1     1     1
     1     1     1
     1     1     1
```

```
all(r(:))
```

1

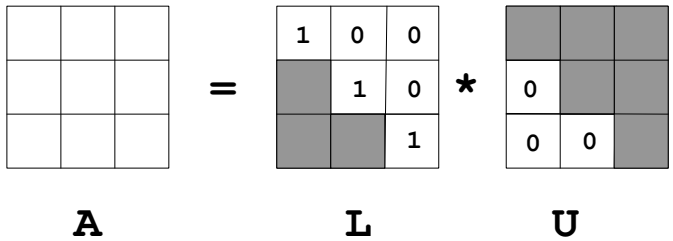
Maple

```
mat := Matrix([[-1,1-3*I,0], [1+3*I,0,-6*I],[0,6*I,1]]);
mat2:= LinearAlgebra:-HermitianTranspose(mat);
LinearAlgebra:-Equal(mat,mat2)
```

true

2.69 Obtain the LU decomposition of a matrix

Problem: Given a matrix A, find matrix L and U such that $LU = A$. The matrix L will have 1 in all the diagonal elements and zeros in the upper triangle. The matrix U will have 0 in the lower triangle as shown in this diagram.



Mathematica

```
Remove["Global`*"]
mat = {{1, -3, 5},
       {2, -4, 7},
       {-1, -2, 1}};

{lu,pivotVector,conditionNumber}=
     LUDecomposition[mat]
```

```
{{{1,-3,5},
 {2,2,-3},
 {-1,-(5/2),-(3/2)}},
{1,2,3},1}
```

```
lower=lu*SparseArray[{i_,j_}/;j<i->1,{3,3}]+
         IdentityMatrix[3]
```

```
{{1,0,0},
 {2,1,0},
{-1,-(5/2),1}}
```

```
upper=lu SparseArray[{i_,j_}/;j>=i->1,{3,3}]
```

```
{{1,-3,5},
 {0,2,-3},
 {0,0,-(3/2)}}
```

```
lower.upper
```

```
{{1,-3,5},
 {2,-4,7},
{-1,-2,1}}
```

Matlab

```
clear all;

A=[1 -3 5;
   2 -4 7;
   -1 -2 1];

[L,U,p]=lu(A)
```

```
L =
    1.0000         0         0
   -0.5000    1.0000         0
    0.5000    0.2500    1.0000

U =
    2.0000   -4.0000    7.0000
         0   -4.0000    4.5000
         0         0    0.3750
p =
     0     1     0
     0     0     1
     1     0     0
```

```
L*U
```

```
ans =
     2    -4     7
    -1    -2     1
     1    -3     5
```

Maple

```
A:=Matrix([[1,-3,5],[2,-4,7],[-1,-2,1]]);
p,l,u:=LinearAlgebra:-LUDecomposition(A);
```

$$\left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -\frac{5}{2} & 1 \end{bmatrix}, \begin{bmatrix} 1 & -3 & 5 \\ 0 & 2 & -3 \\ 0 & 0 & -\frac{3}{2} \end{bmatrix}\right]$$

2.70 Linear convolution of 2 sequences

Problem: Given 2 sequences $x_1[n]$ and $x_2[m]$, determine the linear convolution of the 2 sequences. Assume $x_1 = [1, 2, 3, 4, 5]$ and $x_2 = [3, 4, 5]$.

Mathematica

```
Clear ["Global`*"]
x1 = {1, 2, 3, 4, 5};
x2 = {4, 5, 6};
ListConvolve[x2, x1, {1, -1}, 0]
```

```
{4, 13, 28, 43, 58, 49, 30}
```

Matlab

```
clear all; close all;
x1=[1 2 3 4 5];
x2=[4 5 6];
conv(x2,x1)
```

```
4   13   28   43   58   49   30
```

Maple

In Maple, had to convert lists to Array first to use Convolution. This is not good. The function should have accepted lists also. Maple 2021.

```
x1 := [1, 2, 3, 4, 5];
x2 := [4, 5, 6];
#note, need to convert list to Array. Why?
SignalProcessing:-Convolution(convert(x1,Array),convert(x2,Array))
```

$$\begin{bmatrix} 4.0 & 13.0 & 28.0 & 43.0 & 58.0 & 49.0 & 30.0 \end{bmatrix}$$

2.71 Circular convolution of two sequences

Problem: Given 2 sequences $x_1[n]$ and $x_2[m]$, determine the circular convolution of the 2 sequences.

2.71.1 example 1. The sequences are of equal length

Mathematica

```
x1 = {2, 1, 2, 1};
x2 = {1, 2, 3, 4};
X1 = Chop[Fourier[x1, FourierParameters -> {1, -1} ]];
X2 = Chop[Fourier[x2, FourierParameters -> {1, -1}]];
Re[InverseFourier[X1*X2, FourierParameters -> {1, -1}]]
```

```
{14., 16., 14., 16.}
```

Matlab

```
clear all; close all;
x1=[2 1 2 1];
x2=[1 2 3 4];
X1=fft(x1);
X2=fft(x2);
y=real(ifft(X1.*X2))
```

```
y =
    14    16    14    16
```

Maple

Note: had to figure the correct normalization to get same answer as Matlab. Also, the input has to be type Array. list is not accepted.

```
x1 := Array([2, 1, 2, 1]);
x2 := Array([1,2,3,4]);
X1:=SignalProcessing:-FFT(x1,'normalization'='none');
X2:=SignalProcessing:-FFT(x2,'normalization'='none');
SignalProcessing:-InverseFFT(X1.X2,'normalization'='full');
Re(%)
```

$$\begin{bmatrix} 14.0 & 16.0 & 14.0 & 16.0 \end{bmatrix}$$

2.71.2 example 2. The sequences are of unequal length

Mathematica

```
Clear["Global`*"]
x1 = {1, 2, 3, 4, 5};
x2 = {4, 5, 6};
x2 = Append[x2, {0, 0}] // Flatten;
X1 = Chop[Fourier[x1, FourierParameters -> {1, -1} ]];
X2 = Chop[Fourier[x2, FourierParameters -> {1, -1}]];
Re[InverseFourier[X1*X2, FourierParameters -> {1, -1}]]
```

```
{53., 43., 28., 43., 58.}
```

Matlab

```
x1=[1 2 3 4 5];
x2=[4 5 6];
N=max(length(x1),length(x2));
X1=fft(x1,N);
X2=fft(x2,N);
y=real(ifft(X1.*X2))
```

```
y =
53    43    28    43    58
```

Maple

Padding zeros to make the two list same was a little tricky. Find if there is a better way.

```
x1 := [1,2,3,4,5];
x2 := [4,5,6];

#find max length, and padd the smaller one with zero, so that
#both lists have same length
N:= max(numelems(x1),numelems(x2));
x1:=`if`(numelems(x1)<N,[op(x1),op([0$(N-numelems(x1))])],x1);
x2:=`if`(numelems(x2)<N,[op(x2),op([0$(N-numelems(x2))])],x2);

X1:=SignalProcessing:-FFT(convert(x1,Array),'normalization'='none');
X2:=SignalProcessing:-FFT(convert(x2,Array),'normalization'='none');
SignalProcessing:-InverseFFT(X1.X2,'normalization'='full');
Re(%)
```

$$\begin{bmatrix} 53.0 & 43.0 & 28.0 & 43.0 & 58.0 \end{bmatrix}$$

2.72 Linear convolution of 2 sequences with origin at arbitrary position

For simple case where the 2 sequences are assumed to start at n=0, then this can be done in the time domain using the ListConvolve in Mathematica and using conv in Matlab.

The harder case is when the origin is not located at the first element in the sequence. I.e. one or both of the sequences is not causal.

Mathematica

case 1

Convolve 2 sequences where the origin is assumed to be at the first element in each sequence.

```
x1={1,2,3,4,5};
x2={4,5,6};
ListConvolve[x2,x1,{1,-1},0]
```

```
{4,13,28,43,58,49,30}
```

case 2

Convolve 2 sequences where the origin is located at different location from the first element of the sequence, use DiracDelta function, and the DTFT approach.

Example convolve $x = \{1, 2, 0, 2, 1, 4, 01, 2, 2\}$ with $h = \{1/4, 1/2, 1/4\}$ where the origin of h is under the second element $1/2$.

```
(*Write down the h  and take its DTFT *)
Clear[n,w]
h = 1/4DiscreteDelta[n+1] +
    1/2DiscreteDelta[n]+1/4 DiscreteDelta[n-1];
hTransformed = FourierSequenceTransform[h,n,w]
```

$$\frac{1}{4}e^{-iw}\left(1+e^{iw}\right)^2$$

Write down the x sequence and take its DTFT

```
x = 1 DiscreteDelta[n]+2 DiscreteDelta[n-1]-
    2 DiscreteDelta[n-2]+DiscreteDelta[n-3]+
    4 DiscreteDelta[n-4]-DiscreteDelta[n-5]+
    DiscreteDelta[n-6]+2 DiscreteDelta[n-7];

xTransformed = FourierSequenceTransform[x,n,w]
```

$$e^{-7iw}\left(e^{iw}-e^{2iw}+4e^{3iw}+e^{4iw}-2e^{5iw}+2e^{6iw}+e^{7iw}+2\right)$$

Now multiply the DTFT's and take the inverse

```
z = InverseFourierSequenceTransform[
       xTransformed hTransformed,w,n]
```

$$\begin{cases} -\frac{1}{4} & n == 2 \\ \frac{1}{4} & n == -1 \\ \frac{1}{2} & n == 8 \\ \frac{3}{4} & n == 1 \,||\, n == 5 \,||\, n == 6 \\ 1 & n == 0 \,||\, n == 3 \\ \frac{5}{4} & n == 7 \\ 2 & n == 4 \\ 0 & \text{True} \end{cases}$$

Now convolve z with h again, where z is the convolution of x and h found above. This can be done as follows in one command

```
N@InverseFourierSequenceTransform[
 xTransformed hTransformed hTransformed,w,n]
```

$$\begin{cases} 0.0625 & n == -2. \\ 0.125 & n == 9. \\ 0.3125 & n == 2. \\ 0.375 & n == -1. \\ 0.5625 & n == 1. \,||\, n == 8. \\ 0.75 & n == 0. \\ 0.875 & n == 6. \\ 0.9375 & n == 3. \,||\, n == 7. \\ 1.0625 & n == 5. \\ 1.4375 & n == 4. \\ 0. & \text{True} \end{cases}$$

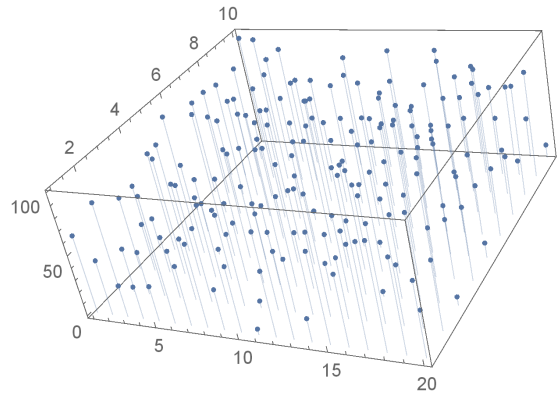
2.73 Visualize a 2D matrix

Problem: Given a 2 dimensional matrix, say $m \times n$, how to visualize its content?

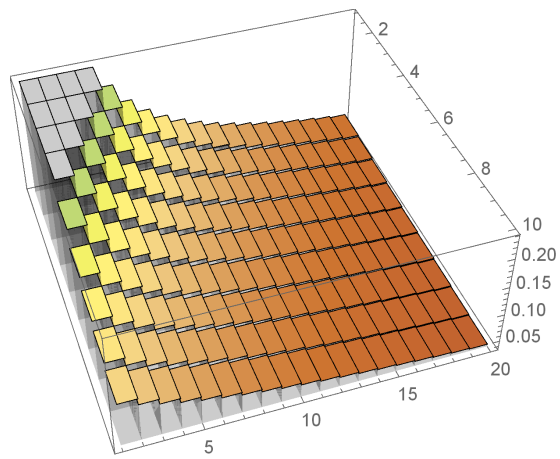
These are some examples showing how to visualize a matrix as a 3D data, where the height is taken as the values of the matrix entries, and the x, y indices as the coordinates.

Mathematica

```
mat = Table[Table[RandomReal[{0,100}],
       {20}],{10}];
ListPointPlot3D[mat,Filling->Axis
                   ,ImageSize->300]
```

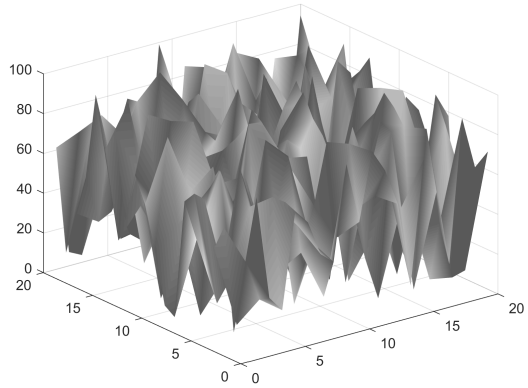



```
mat = HilbertMatrix[{20,10}];
ListPlot3D[mat,Mesh->None,InterpolationOrder->0,
     ColorFunction->"SouthwestColors",
     Filling->Axis,ImageSize->300]
```

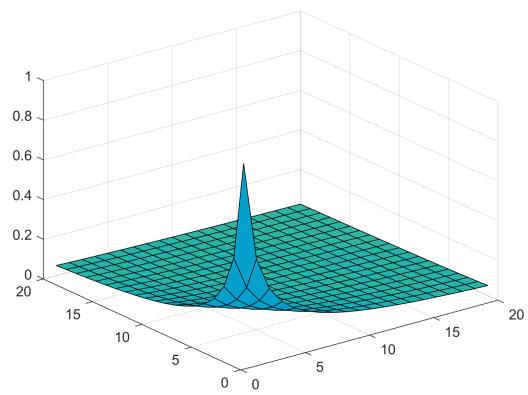


Matlab

```
clear all; close all;

A     = (100).*rand(20,20);
[X,Y] = meshgrid(1:20,1:20);

surfl(X,Y,A);
shading interp
colormap(gray);
```

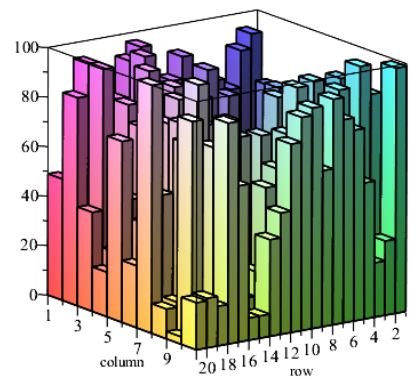



```
figure;
A     = hilb(20);
[X,Y] = meshgrid(1:20,1:20);
surfl(X,Y,A);
```

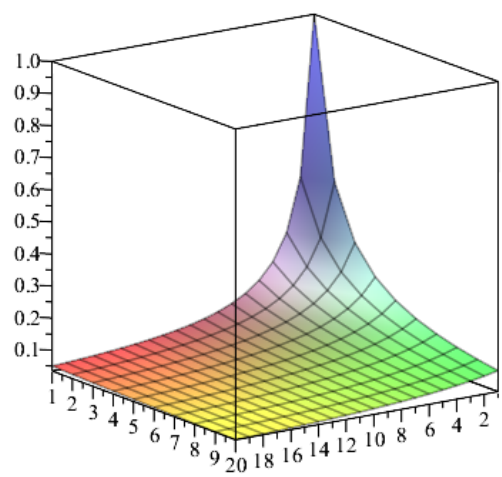


Maple

```
restart;
A := abs(LinearAlgebra[RandomMatrix](20,10,generator=0..100)):
plots[matrixplot](A,heights=histogram);
```




```
A := LinearAlgebra[HilbertMatrix](20,10):
A := convert(A,listlist):
plots[listplot3d](A);
```



2.74 Find the cross correlation between two sequences

Problem: Given

$$A = [0, 0, 2, -1, 3, 7, 1, 2, -3, 0, 0]$$
$$B = [0, 0, 1, -1, 2, -2, 4, 1, -2, 5, 0, 0]$$

Notice that the output sequence generated by Mathematica and Matlab are reversed with respect to each others.

Also, MATLAB uses the length $2N-1$ as the length of cross correlation sequence, which in this example is 23 because N is taken as the length of the larger of the 2 sequences if they are not of equal length which is the case in this example.

In Mathematica, the length of the cross correlation sequence was 22, which is $2N$.

Mathematica

```
Clear["Global`*"];

a={0,0,2,-1,3,7,1,2,-3,0,0};
b={0,0,1,-1,2,-2,4,1,-2,5,0,0};
c=Reverse[ListCorrelate[a,b,{-1,1},0]]
```

```
Out[31]= {0,
          0,
          0,
          0,
          10,
          -9,
          19,
          36,
          -14,
          33,
          0,
          7,
          13,
          -18,
          16,
          -7,
          5,
          -3,
          0,
          0,
          0,
          0}
```

Matlab

In MATLAB use the xcross in the signal processing toolbox

```
clear all; close all;
A=[0,0,2,-1,3,7,1,2,-3,0,0];
B=[0,0,1,-1,2,-2,4,1,-2,5,0,0];
C=xcorr(A,B);
format long
C'
```

```
ans =
   0.000000000000003
   0.000000000000002
  -0.000000000000002
                   0
   9.999999999999998
  -9.000000000000002
  19.000000000000000
  36.000000000000000
 -14.000000000000000
  33.000000000000000
  -0.000000000000002
   6.999999999999998
  13.000000000000000
 -18.000000000000000
  16.000000000000004
  -7.000000000000000
   4.999999999999999
  -2.999999999999998
  -0.000000000000000
   0.000000000000001
   0.000000000000002
  -0.000000000000004
                   0
```

Maple

```
a:=Array([0,0,2,-1,3,7,1,2,-3,0,0]);
b:=Array([0,0,1,-1,2,-2,4,1,-2,5,0,0]);
SignalProcessing:-CrossCorrelation(a,b);
```

$$[7.0, 0.0, 33.0, -14.0, 36.0, 19.0, -9.0, 10.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$$

Not able to find out now why Maple result is different. May be definition used is different, no time now to find out.

2.75 Find orthonormal vectors that span the range of matrix A

Problem: Given the matrix A whose columns represents some vectors, find the set of orthonormal vectors that span the same space as A and verify the result. Let

$$A = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 2 & 3 & 4 \\ 2 & 0 & 2 & 0 \end{bmatrix}$$

Notice that A has rank 2, so we should get no more than 2 vectors in the orthonormal set.

With MATLAB use the **orth(A)** function, With Mathematica, use **{u,s,v}=SingularValueDecomposition[A]** , and since the rank is 2, then the first 2 columns of matrix **u** will give the answer needed (any 2 columns of u will also give a set of orthonormal vectors).

Mathematica

```
Remove["Global`*"];
mat = {{0, 1, 1, 2},
       {1, 2, 3, 4},
       {2, 0, 2, 0}};
r = MatrixRank[mat]
```

2

```
{u,s,v}=SingularValueDecomposition[mat];
orth=N[u[[All,{1,r}]]]
```

```
{{0.378151, -0.308379},
 {0.887675, -0.146825},
{0.262747, 0.939864}}
```

```
Chop[Transpose[orth].orth]
```

```
{{1.,0},
{0,1.}}
```

Matlab

```
clear all;
A=[0 1 1 2
   1 2 3 4
   2 0 2 0];
R=orth(A)
```

```
R =
   -0.3782    0.3084
   -0.8877    0.1468
   -0.2627   -0.9399
```

```
R'*R
```

```
    1.0000    0.0000
    0.0000    1.0000
```

2.76 Solve $Ax = b$ and display the solution

Problem: Solve for x given that $Ax = b$ where

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 3 \end{pmatrix}$$

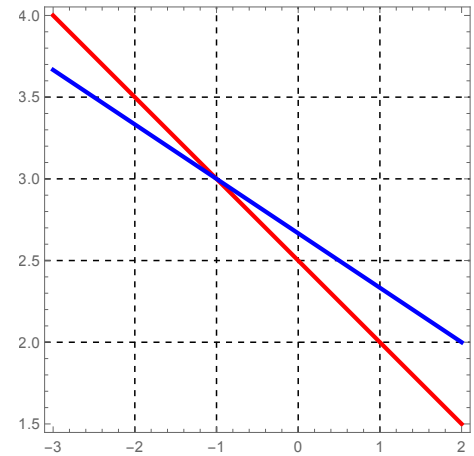
and

$$b = \begin{pmatrix} 5 \\ 8 \end{pmatrix}$$

These 2 equations represent 2 straight lines in a 2D space. An exact solution exist if the 2 lines intersect at a point. The 2 lines are $x + 2y = 5$ and $x + 3y = 8$.

Mathematica

```
Remove["Global`*"];
ContourPlot[{x+2y==5,x+3y==8},
  {x,-3,2},
  {y,1.5,4},
  ContourStyle->{{Red,Thickness[0.01]},
                 {Blue,Thickness[0.01]}},
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  ImageSize->300]
```




```
mat = {{1,2},{1,3}};
b   = {5,8};
x   = LinearSolve[mat,b]
```

```
{-1,3}
```

Matlab

```
A = [1 2;1 3];
b = [5;8];
x = A\b
```

```
x =
    -1
     3
```

2.77 Determine if a set of linear equations A x= b has a solution and what type of solution

Problem: Given a general non homogeneous set of linear equations $Ax = b$ how to test if it has no solution (inconsistent), or one unique solution, or an infinity number of solutions?

The following algorithm summarizes all the cases

```
Let [A|b] be the augmented matrix, where b is appended to A.

Assume A is an M by N matrix. i.e. M equations and N unknowns.
```

```
IF rank A < rank [A|b]  THEN  -- system is inconsistent
   -- NO exact solution exist, but can use least square approximation

   x= A/b  -- Matlab.
   x= PseudoInverse[A].b  -- Mathematica uses SVD to computer pseduoInverse

ELSE -- we must have rank A == rank[A|b]  -- system is consistent
     IF rank(A) == N  -- we have one solution.
        IF M==N -- one unique solution, can use crammer rule.
           x=A/b  -- Matlab. Here we get exact solution from \ operator
           x=LinearSolve[A,b] -- Mathematica
        ELSE  -- infinite solutions, pick one.
           x=A/b  -- Matlab. Here we get exact solution from \ operator
           x=LinearSolve[A,b] -- Mathematica
        END
     ELSE
        -- rank A  < N, i.e. rank deficient, infinite solutions. will pick one
        x=A/b
        x=LinearSolve[A,b]
     END
END
```

Let the system of equations be

$$y = x - 1$$
$$y = 2x + 1$$

So

$$A = \begin{pmatrix} 1 & 1 \\ -2 & 1 \end{pmatrix}$$

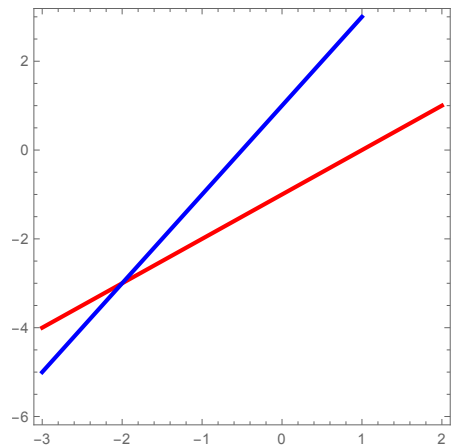
and

$$b = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Mathematica

```
Remove["Global`*"];
ContourPlot[{y==x-1,y==2 x+1} ,
            {x,-3,2},{y,-6,3},
     ContourStyle->{{Red,Thickness[0.01]},
```

```
                    {Blue,Thickness[0.01]}},
     GridLinesStyle->Dashed,
     ImageSize->300]
```




```
a = {{-2,1},{-1,1}};
b = {1,-1};
{nRow, nCol} = Dimensions[a];
aRank=MatrixRank[a]

   2
```

```
abRank=MatrixRank[Insert[a,b,-1]]

   2
```

The above algorithm can now be run as follows

```
If[aRank<abRank,
  Print["System is no consistent, no exact solution"];
  x=PseudoInverse[a].b,
  If[aRank==nCol,
   Print["System is consistent. Exact solution."];
   x=LinearSolve[a,b]
   ,
   Print["consistent, rank deficient,infinite solutions"];
   x=LinearSolve[a,b]
  ]
```

```
];

Print["Solution is x=",x];
```

The output of the above is

```
System is consistent. Exact solution.
Solution is x={-2,-3}
```

Matlab

```
A=[-2 1;-1 1];
b=[1; -1];

[nRow,nCol]=size(A);
aRank=rank(A);
abRank=rank([A b]);

fprintf('A rank=%d, [A b] rank=%d\n',aRank,abRank);
fprintf('Number of unknowns=%d\n',nCol);

x=A\b;

if aRank<abRank
 fprintf('System not consistent. no exact solution\n');
else
 if aRank==nCol
  fprintf('System consistent. Exact solution.\n');
 else
  fprintf('consistent,rank deficient,infinite solutions\n');
 end
end

fprintf('solution is\n');
x
```

Output is

```
A rank=2, [A b] rank=2
Number of unknowns=2
```

```
System is consistent. Exact solution.
solution is
x =
    -2
    -3
```

2.78 Given a set of linear equations automatically generate the matrix A and vector b and solve $Ax = b$

Problem: Given

$$\begin{aligned} 4x + 4y + 2z &= 12 \\ 5x + 6y + 3z &= 7 \\ 9x + 7y + 10z &= 9 \end{aligned}$$

Automatically convert it to the form $Ax = b$ and solve

Mathematica

```
Remove["Global`*"]
eqs = {4x+3y+2z==12,
       5x+6y+3z  ==7,
       9x+7y+10z ==9};

{b,a} = CoefficientArrays[eqs,{x,y,z}];

Normal[a]//MatrixForm
```

$$\begin{pmatrix} 4 & 3 & 2 \\ 5 & 6 & 3 \\ 9 & 7 & 10 \end{pmatrix}$$

```
Normal[b]//MatrixForm
```

$$\{-12, -7, -9\}$$

```
LinearSolve[a,-b]//N
```

```
{6.71429,-2.85714,-3.14286}
```

Maple

```
restart;
eqs:=[4*x+3*y+2*z=12,5*x+6*y+3*z=7,9*x+7*y+10*z=9];
A,b := LinearAlgebra:-GenerateMatrix(eqs,[x,y,z]);
LinearAlgebra:-LinearSolve(A,b)
```

$$\begin{bmatrix} \frac{47}{7} \\ -\frac{20}{7} \\ -\frac{22}{7} \end{bmatrix}$$

2.79 Convert a matrix to row echelon form and to reduced row echelon form

Problem: Given a matrix A, convert it to REF and RREF. Below shows how to convert the matrix A to RREF. To convert to REF (TODO). One reason to convert Matrix A to its row echelon form, is to find the rank of A. If matrix A is a 4×4, and when converted to its row echelon form we find that one of the rows is all zeros, then the rank of A will be 3 and not full rank.

Mathematica

```
Remove["Global`*"]
(mat={{1, 1, -2,  1},
      {3, 2,  4, -4},
      {4, 3,  3, -4}})//MatrixForm
```

$$\begin{pmatrix} 1 & 1 & -2 & 1 \\ 3 & 2 & 4 & -4 \\ 4 & 3 & 3 & -4 \end{pmatrix}$$

```
MatrixForm[RowReduce[mat]]
```

$$\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Matlab

```
clear all;
A=[1 1 -2 1
   3 2 4  -4
   4 3 3  -4];
rref(A)
```

```
ans =
     1     0     0     2
     0     1     0    -3
     0     0     1    -1
```

Maple

```
A:=Matrix([ [1,1,-2,1],[3,2,4,-4],[4,3,3,-4]]);
LinearAlgebra:-ReducedRowEchelonForm(A);
```

$$\begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

2.80 Convert 2D matrix to show the location and values

Given

$$A = \begin{pmatrix} 41 & 45 & 49 & 53 \\ 42 & 46 & 50 & 54 \\ 43 & 47 & 51 & 55 \\ 44 & 48 & 52 & 56 \end{pmatrix}$$

Generate the matrix

$$\begin{pmatrix} 1 & 1 & 41 \\ 2 & 1 & 42 \\ 3 & 1 & 43 \\ 4 & 1 & 44 \\ 1 & 2 & 45 \\ 2 & 2 & 46 \\ 3 & 2 & 47 \\ 4 & 2 & 48 \\ 1 & 3 & 49 \\ 2 & 3 & 50 \\ 3 & 3 & 51 \\ 4 & 3 & 52 \\ 1 & 4 & 53 \\ 2 & 4 & 54 \\ 3 & 4 & 55 \\ 4 & 4 & 56 \end{pmatrix}$$

Which gives at each row, the location and the value in the original matrix.

Mathematica

```
mat = {{41, 45, 49, 53}, {42, 46, 50, 54}, {43, 47, 51, 55}, {44, 48, 52, 56}};
Flatten[MapIndexed[Flatten[{{#2[[2]], #2[[1]]}, #1}] &, Transpose@mat, {2}],1]
```

Another way

```
mat = {{41, 45, 49, 53}, {42, 46, 50, 54}, {43, 47, 51, 55}, {44, 48, 52, 56}};
{nRow, nCol} = Dimensions[mat];
idx = Flatten[Table[{i, j}, {j, nCol}, {i, nRow}], 1]; (*similar to ind2sub*)
val = Extract[mat, idx];
MapThread[Flatten[{#1, #2}] &, {idx, val}] (* similar to [I;J;A(:)']' *)
```

But I think the simplist is to use Table

```
mat = {{41, 45, 49, 53}, {42, 46, 50, 54}, {43, 47, 51, 55}, {44, 48, 52, 56}};
{nRow, nCol} = Dimensions[mat];
Table[{i, j, mat[[i, j]]}, {j, nCol}, {i, nRow}];
Flatten[%, 1]
```

Matlab

```
A = [ 41 45 49 53;
      42 46 50 54;
      43 47 51 55;
      44 48 52 56];
[I,J] = ind2sub(size(A),1:numel(A));
X     = [I;J;A(:)']'
```

gives

```
X =
      1     1    41
      2     1    42
      3     1    43
      4     1    44
      1     2    45
      2     2    46
      3     2    47
      4     2    48
      1     3    49
      2     3    50
      3     3    51
      4     3    52
      1     4    53
      2     4    54
      3     4    55
      4     4    56
```

Maple

```
A := Matrix([[41,45,49,53],[42,46,50,54],[43,47,51,55],[44,48,52,56]]);
```

$$A = \begin{bmatrix} 41 & 42 & 43 & 44 \\ 45 & 46 & 47 & 48 \\ 49 & 50 & 51 & 52 \\ 53 & 54 & 55 & 56 \end{bmatrix}$$

```
[seq(seq([i,j,A(i,j)],j=1..4),i=1..4)]:
indx:=Matrix(%);
```

$$\begin{bmatrix} 1 & 1 & 41 \\ 1 & 2 & 45 \\ 1 & 3 & 49 \\ 1 & 4 & 53 \\ 2 & 1 & 42 \\ 2 & 2 & 46 \\ 2 & 3 & 50 \\ 2 & 4 & 54 \\ 3 & 1 & 43 \\ 3 & 2 & 47 \\ 3 & 3 & 51 \\ 3 & 4 & 55 \\ 4 & 1 & 44 \\ 4 & 2 & 48 \\ 4 & 3 & 52 \\ 4 & 4 & 56 \end{bmatrix}$$

2.81 Find rows in matrix with zeros in them, and then remove the zeros

Given $A = \{\{1,0,9\},\{5,0,6\},\{4,1,9\},\{7,0,11\},\{8,1,2\}\}$

Find rows with zero in middle, and then remove the zeros from these found.

The result should be $\{\{1,9\},\{5,6\},\{7,11\}\}$

Mathematica

Many ways to do this. See this

```
Remove["Global`*"]
A = {{1, 0, 9}, {5, 0, 6}, {4, 1, 9}, {7, 0, 11}, {8, 1, 2}};
Cases[A,{x_,0,y_}:>{x,y}]
```

```
{{1, 9}, {5, 6}, {7, 11}}
```

Maple

```
restart;
A := {{1, 0, 9}, {5, 0, 6}, {4, 1, 9}, {7, 0, 11}, {8, 1, 2}};

#One way is

remove~(has,select(has,A,0),0)

#Another way is

f := x->remove(has,x,0);
f~(select(has,A,0))
```

```
{{1, 9}, {5, 6}, {7, 11}}
```

2.82 How to apply a function to two lists at the same time?

Given list $a = \{1, 2, 3, 4\}$ and list $b = \{5, 6, 7, 8\}$ how to to call function $f(x, y)$ by taking x, y from a, b one a time so that the result gives $f(1, 5), f(2, 6), f(3, 7), f(4, 8)$?

Mathematica

```
Remove["Global`*"]
a = {1, 2, 3, 4};
b = {5, 6, 7, 8};
MapThread[f[#1, #2] &, {a, b}]
```

```
{f[1, 5], f[2, 6], f[3, 7], f[4, 8]}
```

Maple

```
restart;
a:=[1,2,3,4];
b:=[5,6,7,8];
f~(a,b)
```

```
[f(1, 5), f(2, 6), f(3, 7), f(4, 8)]
```

2.83 How to apply a function to two lists are the same time, but with change to entries?

2.83.1 example 1

Given list $a = \{1, 2, 3, 4\}$ and list $b = \{5, 6, 7, 8\}$ how to to call function $f(x, y)$ by taking $\sin(x), \cos(y)$ from a, b one a time so that the result gives

$$f(\sin(1), \cos(5)), f(\sin(2), \cos(6)), f(\sin(3), \cos(7)), f(\sin(4), \cos(8))$$

Mathematica

```
Remove["Global`*"]
a = {1, 2, 3, 4};
b = {5, 6, 7, 8};
MapThread[f[Sin[#1],Cos[#2]] &, {a, b}]
```

```
{f[Sin[1], Cos[5]], f[Sin[2], Cos[6]], f[Sin[3], Cos[7]], f[Sin[4], Cos[8]]}
```

Maple

```
restart;
a:=[1,2,3,4];
b:=[5,6,7,8];
f~(sin~(a),cos~(b))
```

```
[f(sin(1), cos(5)), f(sin(2), cos(6)), f(sin(3), cos(7)), f(sin(4), cos(8))]
```

2.83.2 example 2

Given list $a = \{1, 2, 3, 4\}$ and list $b = \{5, 6, 7, 8\}$ how to to call function $f(2x + x^2 + \sin(x), 2 + \cos(y))$ by taking x, y from a, b one a time so that the result gives

$$\{f(3+\sin(1), 5+\cos(5)), f(8+\sin(2), 6+\cos(6)), f(15+\sin(3), 7+\cos(7)), f(24+\sin(4), 8+\cos(8))\}$$

Mathematica

```
Remove["Global`*"]
a = {1, 2, 3, 4};
b = {5, 6, 7, 8};
MapThread[f[2*#1 + #1^2 + Sin[#1], #2 + Cos[#2]] &, {a, b}]
```

```
{f[3+Sin[1],5+Cos[5]],f[8+Sin[2],6+Cos[6]],f[15+Sin[3],7+Cos[7]],f[24+Sin[4],8+Cos[8]]}
```

Maple

In Maple, since it does not have slot numbers # to use, it is better to make a function on the fly, which does the exact same thing.

```
restart;
a:=[1,2,3,4];
b:=[5,6,7,8];

f~( (x->2*x+x^2+sin(x))~(a), (x->2+cos(x)) ~(b))
```

```
[f(3 + sin(1), 2 + cos(5)), f(8 + sin(2), 2 + cos(6)), f(15 + sin(3), 2 + cos(7)), f(24 + si
```

2.84 How to select all primes numbers from a list?

Given list of numbers from 1 to 100, select the prime numbers.

Mathematica

```
Remove["Global`*"]
lst = Range[1, 100];
Select[lst, PrimeQ]
```

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97}
```

Maple

```
restart;
lst:=[seq(i,i=1..100)];
result:=select[flatten](isprime,lst);

#or. using Array instead of list
lst:=Array(0..99,x->x+1): #generate numbers from 1 to 100
select[flatten](isprime,lst);
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

2.85 How to collect result inside a loop when number of interation is not known in advance?

Lets say we want to collect result obtained inside loop, but do not know in advance how many iteration we need.

In Mathematica, `Sow` and `Reap` are used. In Maple, an `Array` can be used or a queue or a table.

Mathematica

```
Remove["Global`*"]
result = First@Last@Reap@Do[
     Sow[n],
     {n, 1, 10}
     ];
result
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

Maple

```
restart;
result :=Array([]):
for n from 1 to 10 do
    result(n):=n;
od:
result(1..10)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

2.86 How flip an array around?

Given $A = [1, 2, 3, 4, 5]$ change it to $A = [5, 4, 3, 2, 1]$

Mathematica

```
a = {1, 2, 3, 4, 5};
Reverse[a]
```

```
{5, 4, 3, 2, 1}
```

Matlab

```
A=[1,2,3,4,5];
flip(A)
```

```
ans =
 5     4     3     2     1
```

Maple

```
A:=Array([1,2,3,4,5]);
ArrayTools:-Reverse(A);
```

```
[5, 4, 3, 2, 1]
```

2.87 How to divide each element by its position in a list?

Given $A = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]$ change it to

$$[2, \frac{3}{2}, \frac{5}{3}, \frac{7}{4}, \frac{11}{5}, \frac{13}{6}, \frac{17}{7}, \frac{19}{8}, \frac{23}{9}, \frac{29}{10}]$$

2.87.1 Mathematica

```
A = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
MapIndexed[(#1/First[#2]) &, A]
```

$$\left\{2, \frac{3}{2}, \frac{5}{3}, \frac{7}{4}, \frac{11}{5}, \frac{13}{6}, \frac{17}{7}, \frac{19}{8}, \frac{23}{9}, \frac{29}{10}\right\}$$

2.87.2 Maple

```
A:= [2, 3, 5, 7, 11, 13, 17, 19, 23, 29];
((x,y)->x/y)~(A,[seq(i,i=1..numelems(A))])

#or simpler might be to map the divsion operator directly

`/` ~ (A,[seq(i,i=1..numelems(A))])
```

$$[2, \frac{3}{2}, \frac{5}{3}, \frac{7}{4}, \frac{11}{5}, \frac{13}{6}, \frac{17}{7}, \frac{19}{8}, \frac{23}{9}, \frac{29}{10}]$$

2.87.3 Matlab

```
A=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29];
A./(1:length(A))
```

```
2.0000    1.5000    1.6667    1.7500    2.2000    2.1667    2.4286    2.3750    2.5556    2.
```

2.88 How to use GramSchmidt to find a set of orthonomal vectors?

Given set of linearly independent vectors $V_1 = [3, 0, 0], V_2 = [0, 1, 2], V_3 = [0, 2, 5]$, use GramSchmidt to find 3 orthonormal vectors from this set.

2.88.1 Mathematica

```
v1={3,0,0};
v2={0,1,2};
v3={0,2,5}
Orthogonalize[{v1, v2,v3}, Method -> "GramSchmidt"]
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ 0 & -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{pmatrix}$$

2.88.2 Maple

```
restart;
V1:=Vector([3,0,0]);
V2:=Vector([0,1,2]);
V3:=Vector([0,2,5]);
Student:-LinearAlgebra:-GramSchmidt([V1,V2,V3])
```

$$\left[\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ \frac{\sqrt{5}}{5} \\ \frac{2\sqrt{5}}{5} \end{bmatrix}, \begin{bmatrix} 0 \\ -\frac{2\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} \end{bmatrix}\right]$$

CHAPTER 3

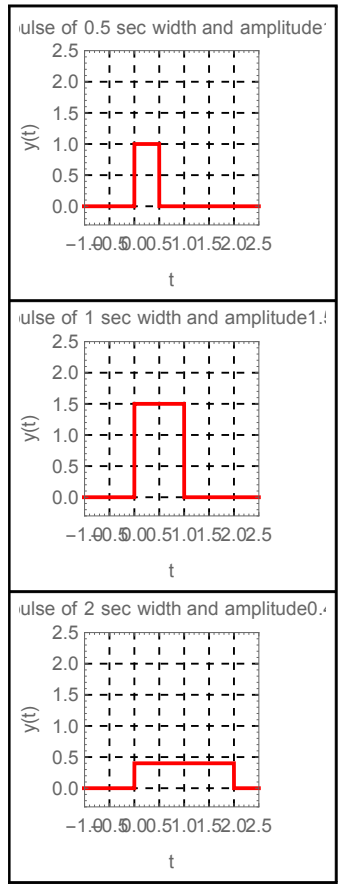
SIGNAL PROCESSING, IMAGE PROCESSING, GRAPHICS, RANDOM NUMBERS

3.1 Generate and plot one pulse signal of different width and amplitude

Problem: Generate one signal of some width and height.

Mathematica

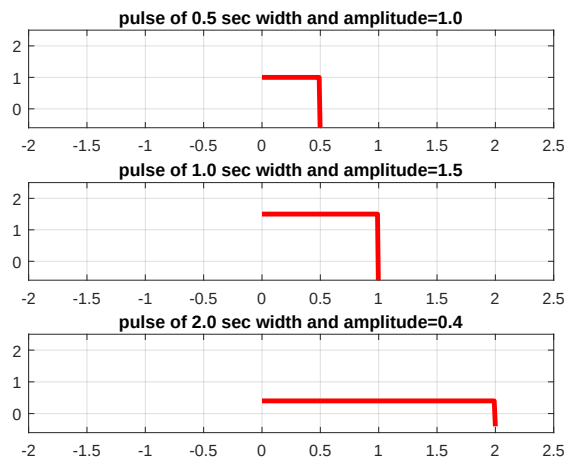
```
Remove["Global`*"]
f[x_,w_,h_]:=If[0<=x<w,
      h*If[x-IntegerPart[x]<=0.5,
      SquareWave[x],
      -SquareWave[x]
      ],
      0]
w = {0.5,1,2};    (*pulse width *)
h = {1,1.5,0.4}; (*pulse height *)
plots = MapThread[Plot[
          f[x,#1,#2],{x,-2,2.5},
            PlotRange->{{-1,2.5},{-0.3,2.5}},
            Frame->True,
            Axes->False,
            GridLines->Automatic,
            GridLinesStyle->Dashed,
            PlotStyle->{Thick,Red},
            FrameLabel->{{"y(t)",None},
    {"t","pulse of "<>ToString[#1]<>
    " sec width and amplitude"<>
    ToString[#2]}},
            ImageSize -> 160,
            AspectRatio->1]&,{w,h}];

Grid[{plots},Frame->All]
```



Matlab

```
clear all; close all;
f = [1 0.5 0.25];
w = [0.5 1 2];
h = [1 1.5 0.4];

figure;

for i=1:3
  t=0:0.01:w(i);
  y=h(i)*square(2*pi*f(i)*t);
```

```
  subplot(3,1,i);
  plot(t,y,'r','LineWidth',3);
  grid on;
  xlim([-2 2.5]);
  ylim([-.6 2.5]);
  title(sprintf...
 ('pulse of %2.1f sec width and amplitude=%2.1f',...
  w(i),h(i)));

end
```

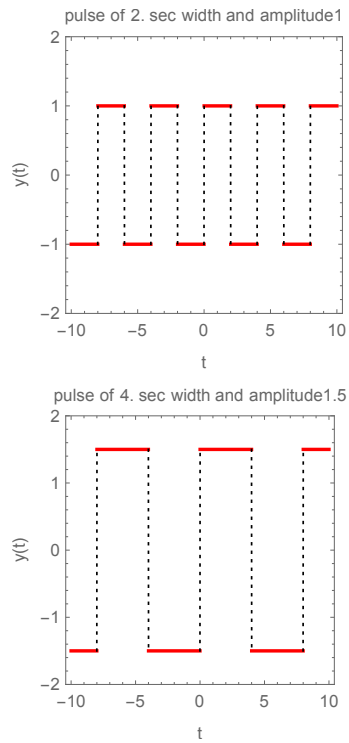


3.2 Generate and plot train of pulses of different width and amplitude

Problem: Generate a pulse train of pulses of certain width and height.

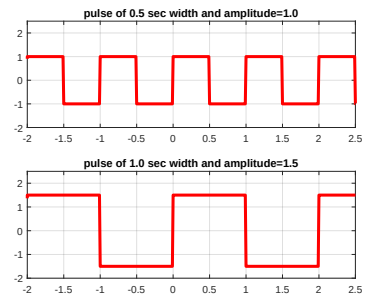
Mathematica

```
Remove["Global`*"]

w = {0.25,0.125};
h = {1,1.5};
plots = MapThread[Plot[#2 SquareWave[#1 x],
         {x,-10,10},
  PlotRange->{All,{-2,2}},
  Frame->True,
  Axes->False,
  PlotStyle->{Thick,Red},
  FrameLabel->{{"y(t)",None},
   {"t","pulse of "<>ToString[1/(2*#1)]<>
   " sec width and amplitude"<>ToString[#2]}},
  ImageSize->200,
  AspectRatio->1,
  ExclusionsStyle->Dotted]&,{w,h}];

Grid[{plots},Frame->All]
```



Matlab

```
clear all; close all;
f = [1 0.5];
h = [1 1.5];
t = -4:0.01:4;

figure;

for i=1:2
  y = h(i)*square(2*pi*f(i)*t);
  subplot(2,1,i);
  plot(t,y,'r','LineWidth',3);
  grid on;
  xlim([-2 2.5]);
  ylim([-2 2.5]);
  title(sprintf...
 ('pulse of %2.1f sec width and amplitude=%2.1f',..
   1/(2*f(i)),h(i)));
end
```



3.3 Find the discrete Fourier transform of a real sequence of numbers

Given the sequence of numbers $x(n) = [1, 2, 3]$, Find $X(k) = \sum_{m=0}^{N-1} x(m) e^{-j\frac{2\pi}{N}mk}$ where N is the length of the data sequence $x(m)$ and $k = 0 \cdots N-1$

Mathematica

```
Chop[Fourier[{1, 2, 3},
      FourierParameters ->{1, -1}]]
```

```
{6.,
-1.5 + 0.8660254037844386*I,
-1.5 - 0.8660254037844386*I}
```

Matlab

```
fft([1,2,3])'
```

```
ans =
    6.0000
   -1.5000 - 0.8660i
   -1.5000 + 0.8660i
```

Maple

Maple need an Array for input, not list, so have to convert this is little strange

```
lst:=[1,2,3];
SignalProcessing[FFT](convert(lst,Array),
          normalization=none );
```

```
[6.+0.*I,
 -1.5+.866025403784439*I,
 -1.5-.866025403784439*I]
```

Ada I posted the code below on comp.lang.ada on June 8,2010. Thanks to Ludovic Brenta for making improvement to the Ada code.

```
--
-- dtf.adb, compiled with GNAT 4.3.4 20090804 (release) 1
-- under CYGWIN 1.7.5
-- $ gnatmake dft.adb
-- gcc -c dft.adb
-- gnatbind -x dft.ali
-- gnatlink dft.ali
-- $ ./dft.exe
-- ( 6.00000E+00, 0.00000E+00)
-- (-1.50000E+00, 8.66026E-01)
-- (-1.50000E+00,-8.66025E-01)

with Ada.Text_IO; use Ada.Text_IO;
with Ada.Numerics.Complex_Types; use  Ada.Numerics.Complex_Types;

with Ada.Numerics; use  Ada.Numerics;

with Ada.Numerics.Complex_Elementary_Functions;
use  Ada.Numerics.Complex_Elementary_Functions;
with Ada.Complex_Text_IO; use Ada.Complex_Text_IO;

procedure dft is
     N : constant := 3; -- named number, no conversion to Float needed
     X : array(0 .. N-1) of Complex  := (others=>(0.0,0.0));
     data : constant array(0 .. N-1) of float :=(1.0,2.0,3.0);
     Two_Pi_Over_N : constant := 2 * Pi / N;
      -- named number, outside the loop, like in ARM 3.3.2(9)
begin
     FOR k in X'range LOOP
         FOR m in data'range LOOP
             X(k) := X(k) + data(m)*exp(-J*Two_Pi_Over_N * float(m*k));
         END LOOP;
         put(X(k)); new_line;
     END LOOP;
end dft;
```

Fortran

Thanks to Vincent Lafage for making improvment to the Fortran code.

```
 ! dtf.f90, compiled with GCC 4.3.4
! under CYGWIN 1.7.5
!  $ gfortran -Wall -Wsurprising -Wconversion dft.f90
!  $ ./a.exe
! (  6.0000000    ,  0.0000000    )
! ( -1.4999999    , 0.86602557    )
! ( -1.5000005    ,-0.86602497    )
!  $

PROGRAM dft

  IMPLICIT NONE

  INTEGER, parameter :: N = 3
  COMPLEX, parameter :: J =(0.0,1.0)
  REAL,    parameter :: Pi = ACOS(-1.0)
  INTEGER                   :: k,m
  COMPLEX, dimension(0:N-1) :: X
  REAL,    dimension(0:N-1) :: data=(/1.0,2.0,3.0/)
  REAL,    parameter        :: Two_Pi_Over_N = 2.0*Pi/real(N)

DO k = lbound(X, 1), ubound(X, 1)
   X(k)=(0.0,0.0)
   DO m = lbound(data, 1), ubound(data, 1)
      X(k) = X(k) + complex(data(m),0.0)                    &
             * EXP(-J*complex(Two_Pi_Over_N*real(m*k),0.0))
   END DO
   print *,X(k)
END DO

END PROGRAM dft
```

3.4 Generate uniform distributed random numbers

3.4.1 How to generate 5 uniform distributed random numbers from 0 to 1?

Mathematica

```
SeedRandom[1];
Table[RandomVariate[
   UniformDistribution[{0,1}]],{5}]
```

```
{0.817389,
0.11142,
0.789526,
0.187803,
0.241361}
```

Matlab

```
rand(5,1)
```

```
0.814723686393179
0.905791937075619
0.126986816293506
0.913375856139019
0.632359246225410
```

Fortran

```
 program t3
implicit none
real :: x(5)

 CALL RANDOM_SEED()
 CALL random_number(x)
 print *,x

end program
```

compile and run

```
$ gfortran -std=f95 -Wextra -Wall -pedantic -funroll-loops
  -ftree-vectorize -march=native  -Wsurprising -Wconversion
  t3.f90  /usr/lib/liblapack.a /usr/lib/libblas.a
$ ./a.exe
  0.99755955      0.56682467      0.96591532      0.74792767      0.36739087
```

3.4.2 How to generate 5 random numbers from a to b?

Generate uniform numbers from a to b, say a=-2 and b=5

Mathematica

```
SeedRandom[1];
Table[RandomVariate[
    UniformDistribution[{-2,5}]],{5}]
```

```
{3.72173,
-1.22006,
 3.52668,
-0.685378,
-0.310473}
```

Matlab

```
-2 + (5+2)*rand(5,1)
```

```
-1.317217165004133
-0.050512467930661
 1.828170634434887
 4.702547848040084
 4.754219746394936
```

Fortran

```
 program t3_2
implicit none
integer ::i
real, parameter :: a=-1.0,b=1.0
real :: x(5)

 CALL RANDOM_SEED()
 DO i=1,2
    CALL random_number(x)
    x = a+(b-a)*x
    print *,x
 END DO

end program
```

compile and run

```
$ gfortran -std=f95 -Wextra -Wall -pedantic -funroll-loops
  -ftree-vectorize -march=native  -Wsurprising -Wconversion
  t3_2.f90  /usr/lib/liblapack.a /usr/lib/libblas.a
$ ./a.exe
  0.99511909      0.13364935      0.93183064      0.49585533     -0.26521826
 -3.87262106E-02 -0.85249150     -0.98928964     -0.30583751     -0.31551242
$
```

3.4.3 How to generate MATRIX of random numbers from a to b?

Let $a = -2$ and $b = 5$, matrix of size 5 by 5

Mathematica

```
SeedRandom[1];
Table[RandomVariate[
   UniformDistribution[{-2,5}]],{5},{5}]
```

```
{{3.72173,-1.22006,3.52668,-0.685378,-0.310473},
 {-1.53983,1.79573,-0.381918,0.772043,2.90332},
```

```
 {-0.517218,3.2406,0.959955,-0.267537,4.8402},
 {3.77614,4.47693,2.04639,0.0500882,-0.543643},
 {2.06332,-1.09825,0.144992,2.98408,0.734073}}
```

Matlab

```
-2 + (5+2)*rand(5,5)
```

```
ans =
 3.7642    1.0712    1.4284   -0.0678    1.4885
 2.8638    0.6709    1.1191    2.7579    4.7182
 0.2197    3.3586    2.5242    2.5857    0.3827
 4.6516    3.5664    2.9656   -0.8617    2.0969
 -1.7589   -0.6919    3.2828   -1.1670   -0.4333
```

Fortran

```
 program t3_3
implicit none
integer ::i
real, parameter :: a=-1.0,b=1.0
real :: x(5,5)

 CALL RANDOM_SEED()
 DO i=1,2
    CALL random_number(x)
    x = a+(b-a)*x
    CALL write(x)
 END DO

 !--- internal functions below ------
     contains
       SUBROUTINE write(A)
       implicit none
       REAL, DIMENSION(:,:) :: A
       integer :: i,j

       WRITE(*,*)
       DO i = lbound(A,1), ubound(A,1)
          WRITE(*,*) (A(i,j), j = lbound(A,2), ubound(A,2))
       END DO

  END SUBROUTINE write

end program
```

compile and run

```
$ gfortran -std=f95 -Wextra -Wall -pedantic -funroll-loops
    -ftree-vectorize -march=native  -Wsurprising -Wconversion
    t3_3.f90  /usr/lib/liblapack.a /usr/lib/libblas.a

$ ./a.exe
  0.99511909     -3.87262106E-02 -0.56409657      0.32386434      0.71138477
  0.13364935     -0.85249150     -0.73367929     -0.96778345     -0.19742620
  0.93183064     -0.98928964      0.80104899      0.30170965     -0.58625138
  0.49585533     -0.30583751     -0.22646809      0.29281759      0.93707883
```

```
-0.26521826     -0.31551242     -0.10903549     -0.35402548      0.19679904

 0.34596145      0.21138644      0.22462976      0.31809497      0.45771694
-8.62354040E-02  0.43809581      0.95732045      0.10801017     -0.19508958
-0.33996975      0.79466915      0.99828446      0.95552015      0.85725522
-0.79923415      0.31645823     -0.48640406      0.80384660     -0.70432973
 0.51090658     -0.69856644      0.10173070      0.31584930      0.34905851
```

3.5 Obtain Fourier Series coefficients for a periodic function

3.5.1 Example 1

Given a continuous time function $x(t) = \sin\left(\frac{2\pi}{T_0}t\right)$, find its Fourier coefficients c_n, where (Using the default definition)

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\omega_0 nt}$$

and

$$c_n = \frac{1}{T_0}\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t)e^{-j\omega_0 nt}dt$$

Where T_0 is the fundamental period of $x(t)$ and $\omega_0 = \frac{2\pi}{T_0}$.

Notice that in Physics, sometimes the following defintions are used instead of the above

$$x(t) = \frac{1}{\sqrt{T_0}}\sum_{n=-\infty}^{\infty} c_n e^{j\omega_0 nt}$$

and

$$c_n = \frac{1}{\sqrt{T_0}}\int_{-\frac{T_0}{2}}^{\frac{T_0}{2}} x(t)e^{-j\omega_0 nt}dt$$

3.5.1.1 Mathematica

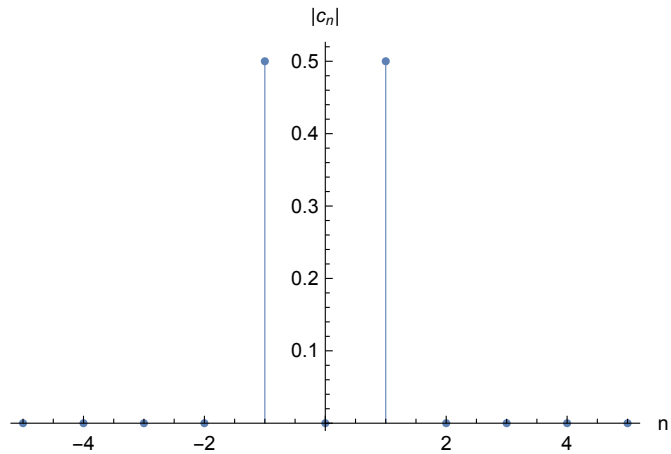
```
Clear[T0, w0]
sol = FourierSeries[Sin[2 Pi/T0 t], t, 3, FourierParameters -> {1, 2 Pi/T0}]
```

$$\frac{1}{2}ie^{-\frac{2i\pi t}{\mathrm{T0}}}-\frac{1}{2}ie^{\frac{2i\pi t}{\mathrm{T0}}}$$

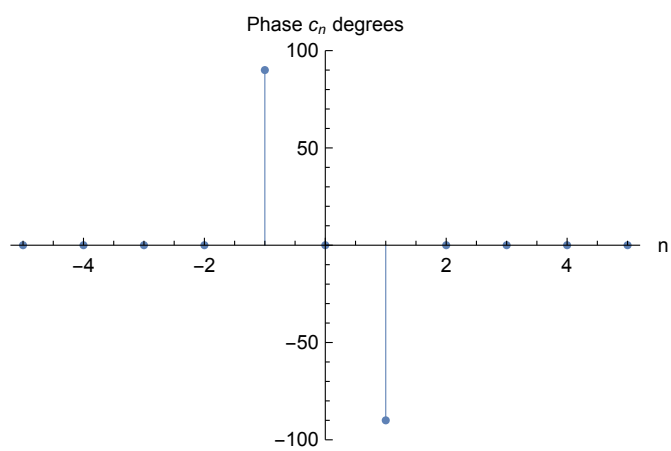
```
data = Table[{i,
  FourierCoefficient[Sin[2 Pi/T0 t], t, i,
  FourierParameters -> {1, 2 Pi/T0}]}, {i, -5, 5}];
  head = {"n", "cn"};
Grid[Insert[data, head, 1], Frame -> All]
```

n	c_n
-5	0
-4	0
-3	0
-2	0
-1	$\frac{i}{2}$
0	0
1	$-\frac{i}{2}$
2	0
3	0
4	0
5	0

```
mag = data;
mag[[All,2]] = Map[Abs[#]&,data[[All,2]]];
ListPlot[mag,AxesOrigin->{0,0},
         Filling->Axis,
         FillingStyle->{{Thick,Red}},
         AxesLabel->{"n","|Subscript[c, n]|"}]
```




```
phase = data;
phase[[All,2]]=Map[Arg[#]&,
           data[[All,2]]]* 180/Pi;
ListPlot[phase,AxesOrigin->{0,0},
  Filling->Axis,
  FillingStyle->{{Thick,Red}},
  AxesLabel->{"n","Phase Subscript[c, n] degrees"},
  ImageSize->300]
```

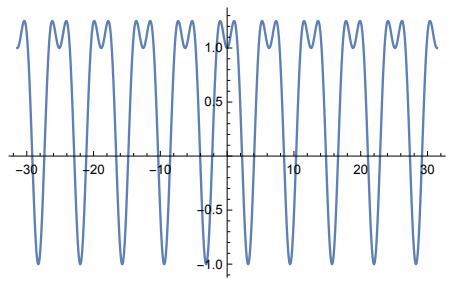


3.5.2 Example 2

Find Fourier Series for $f(t) = \cos(\omega_0 t) + \sin^2(\omega_0 t)$

3.5.2.1 Mathematica

```
T0 = 2 Pi;
w0 = (2 Pi)/T0;
f[t_] := Cos[w0 t] + Sin[w0 t]^2;
Plot[f[t], {t, -10 Pi, 10 Pi}, PlotRange -> All, ImageSize -> 300]
```



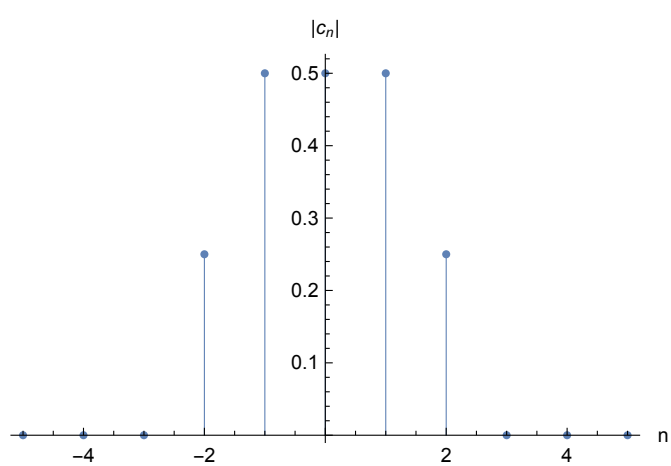

```
sol = FourierSeries[f[t],t,3, FourierParameters->{1,(2 Pi)/T0}]
```

$$\frac{e^{-it}}{2} + \frac{e^{it}}{2} - \frac{1}{4}e^{-2it} - \frac{1}{4}e^{2it} + \frac{1}{2}$$

```
data = Table[{i, FourierCoefficient[f[t],t,i, FourierParameters->{1,2 Pi/T0}]},{i,-5,5}];

head = {"n","cn"};
Grid[Insert[data,head,1],Frame->All]
```

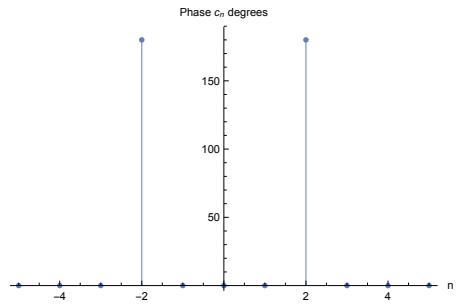
n	c_n
−5	0
−4	0
−3	0
−2	$-\frac{1}{4}$
−1	$\frac{1}{2}$
0	$\frac{1}{2}$
1	$\frac{1}{2}$
2	$-\frac{1}{4}$
3	0
4	0
5	0

```
mag = data;
mag[[All,2]] = Map[Abs[#]&,data[[All,2]]];
ListPlot[mag,AxesOrigin->{0,0},
 Filling->Axis,
 FillingStyle->{{Thick,Red}},
 AxesLabel->{"n","|Subscript[c, n]|"},
 ImageSize->300]
```



Plot the phase

```
phase = data;
phase[[All, 2]] = Map[Arg[#] &,
                  data[[All, 2]]]* 180/Pi;
ListPlot[phase, AxesOrigin -> {0, 0},
 Filling -> Axis,
 FillingStyle -> {{Thick, Red}},
 AxesLabel -> {"n", "Phase cn degrees"}]
```

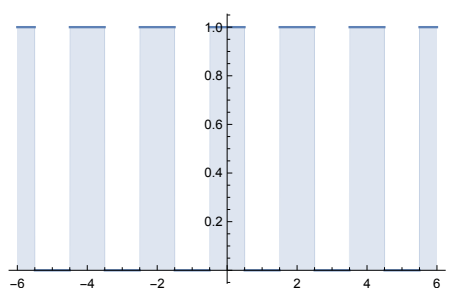


3.5.3 Example 3

Find Fourier series for periodic square wave

3.5.3.1 Mathematica

```
f[x_] := SquareWave[{0,1},(x+.5)/2] ;
Plot[f[x],{x,-6,6},Filling->Axis, ImageSize->300]
```



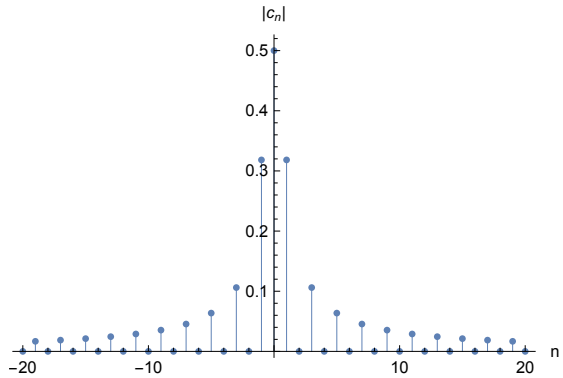

```
T0   = 2;
sol = Chop[FourierSeries[f[t],t,6,  FourierParameters->{1,(2 Pi)/T0}]]
```

$$0.5+0.31831e^{-i\pi t}+0.31831e^{i\pi t}-0.106103e^{-3i\pi t}-0.106103e^{3i\pi t}+0.063662e^{-5i\pi t}+0.063662e^{5i\pi t}$$

```
data = Chop[Table[{i,FourierCoefficient[f[t],t,i, FourierParameters->{1,2 Pi/T}]},{i,-8,8}]]

head = {"n","cn"};
Grid[Insert[data,head,1],Frame->All]
```

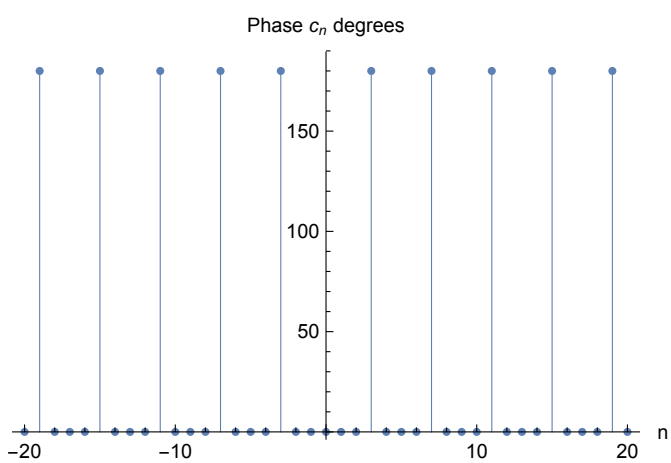
n	c_n
-8	0
-7	-0.0454728
-6	0
-5	0.063662
-4	0
-3	-0.106103
-2	0
-1	0.31831
0	0.5
1	0.31831
2	0
3	-0.106103
4	0
5	0.063662
6	0
7	-0.0454728
8	0

```
data=Table[{i,FourierCoefficient[f[t],t,i,  FourierParameters->{1,2 Pi/T}]},{i,-20,20}];
mag = data;
mag[[All,2]]=Map[Abs[#]&,data[[All,2]]];
ListPlot[mag,AxesOrigin->{0,0},
  Filling->Axis,
  FillingStyle->{{Thick,Red}},
  PlotRange->All,
  AxesLabel->{"n","|Subscript[c, n]|"},
  ImageSize->300]
```



Show phase

```
phase = data;
phase[[All,2]]=Map[Arg[#]&,data[[All,2]]]* 180/Pi;
ListPlot[phase,AxesOrigin->{0,0},
 Filling->Axis,
 FillingStyle->{{Thick,Red}},
 AxesLabel->{"n","Phase cn degrees"},
 ImageSize->300]
```



3.6 Obtain Fourier Series approximation

3.6.1 Example 1

Obtain Fourier Series approximation of $f(x) = e^{-|x|}$ for $-1 < x < 1$

3.6.1.1 Maple

```
restart;
f:=x->exp(-abs(x));
f_approx:=OrthogonalExpansions:-FourierSeries(f(x),x=-1..1,infinity ):
f_approx:=subs(i=n,f_approx);
```

$$1 - \mathrm{e}^{-1} + \sum_{n=1}^{\infty} \left(-\frac{2((-1)^n \mathrm{e}^{-1} - 1)\cos(\pi n x)}{\pi^2 n^2 + 1} \right)$$

3.6.1.2 Mathematica

Mathematica does not have a buildin function to give general series expression as the above with Maple. There is a user written package and answer here `https://mathematica.stackexchange.com/questions/149468/a-more-convenient-fourier-series` which provides this.

In Mathematica it is possible to obtain the terms using the command `FourierSeries` . For example the terms $n = 0, n = -1, n = 1$ can be obtained using

```
expr = Exp[-Abs[x]];
FourierSeries[expr, x, 1, FourierParameters -> {1, Pi}]
```

$$\frac{(1+e)e^{-i\pi x}}{e + e\pi^2} + \frac{(1+e)e^{i\pi x}}{e + e\pi^2} + \frac{e-1}{e}$$

see `https://reference.wolfram.com/language/ref/FourierSeries.html` for definitions of `FourierParameters` used above.

3.6.2 Example 2

Obtain Fourier Series approximation of

$$f(x) = \begin{cases} \frac{2xh}{L} & 0 \leq x \leq \frac{L}{2} \\ \frac{2h(L-x)}{L} & \frac{L}{2} \leq x \leq L \end{cases}$$

For $0 < x < L$

3.6.2.1 Maple

```
restart;
f:=x->piecewise(0<x and x<L/2,2*x*h/L, L/2<x and x<L, 2*h*(L-x)/L);
f_approx:=OrthogonalExpansions:-FourierSeries(f(x),x=0..L,infinity ):
f_approx:=subs(i=n,f_approx):
simplify(%) assuming L>0
```

$$2\left(\sum_{n=1}^{\infty} \frac{h((-1)^n - 1)\cos\left(\frac{2\pi n x}{L}\right)}{\pi^2 n^2}\right) + \frac{h}{2}$$

3.6.3 Example 3

Obtain Fourier Series approximation of $\cosh x$ for $-1 < x < 1$.

3.6.3.1 Maple

```
restart;
f:=x->cosh(x);
f_approx:=OrthogonalExpansions:-FourierSeries(f(x),x=-1..1,infinity ):
f_approx:=subs(i=n,f_approx);
convert(%,trig);
```

$$\sinh(1) + \left(\sum_{n=1}^{\infty} \frac{2\sinh(1)(-1)^n \cos(\pi n x)}{\pi^2 n^2 + 1}\right)$$

3.7 Determine and plot the CTFT (continuous time Fourier Transform) for continuous time function

Plot the CTFT $X(\omega)$ of $x(t) = 3\sin(t)$. By definition

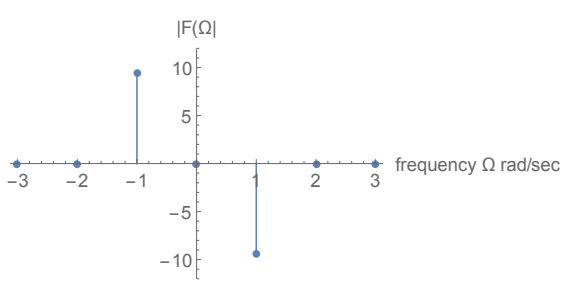
$$X(\omega) = \int_{t=-\infty}^{\infty} x(t)e^{-i\omega t}\,dt$$

Mathematica

```
Clear["Global`*"];
f = 3*Sin[t];
y = FourierTransform[f,t,w,FourierParameters->{1, -1}]
```

```
Out[138]= -3 I Pi DiracDelta[-1+w]+3 I Pi DiracDelta[1+w]
```

```
tab = Table[{k,y/.w->k},{k,-3,3}]
tab = tab/.DiracDelta[0]->1
tab[[All,2]]=Map[Sign[Im[#]]Abs[#]&,tab[[All,2]]]

ListPlot[tab,PlotStyle->AbsolutePointSize[5],
  AxesLabel->{"frequency \[CapitalOmega] rad/sec",
  "|F(\[CapitalOmega]|"},
  PlotRange->{All,{-12,12}},
  Filling->Axis,
  FillingStyle->{{Thick,Red}}]
```



Matlab

```
syms t;
```

```
F=fourier(3*sin(t))
```

```
F =
-pi*(dirac(w-1)-dirac(w+1))*3*i
```

Need to do the plot.

Maple

```
restart;
transform := expand(inttrans:-fourier(3*sin(t), t, w))
```

gives

```
transform := 3*I*Pi*Dirac(w + 1) - 3*I*Pi*Dirac(w - 1)
```

3.8 Determine the DTFT (Discrete time Fourier Transform) for discrete time function

Find DTFT $X(\Omega)$ of $x[n] = \sin(\frac{\pi n}{8})$. By definition

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-i\Omega n}$$

Mathematica

```
Clear["Global`*"];
x[n_] := Sin[(Pi*n)/8];
X = FourierSequenceTransform[x[n], n, w, FourierParameters -> {1, 1}] // Simplify
```

Which gives

```
(-(1/2))*I*(DiracComb[(Pi - 8*w)/(16*Pi)] - DiracComb[(Pi + 8*w)/(16*Pi)])
```

3.9 Determine the Inverse DTFT (Discrete time Fourier Transform)

Find $x[n]$, give its DTFT $X(\Omega)$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\Omega) e^{i\Omega n} \, d\Omega$$

Mathematica

```
Clear["Global`*"];
x[n_] := Sin[(Pi*n)/8];
X = FourierSequenceTransform[x[n], n, w, FourierParameters -> {1, 1}];
InverseFourierSequenceTransform[X, w, n]
```

Which gives

```
Sin[(Pi*n)/8]
```

3.10 Use FFT to display the power spectrum of the content of an audio wav file

Problem: download a wav file and display the frequency spectrum of the audio signal using FFT. The Matlab example was based on Matheworks tech note 1702.

3.10.1 Mathematica

```
Remove["Global`*"];
fname = "stereo.wav";
SetDirectory[NotebookDirectory[]];
ele = Import[fname,"Elements"]
```

```
{AudioChannels,AudioEncoding,
 Data,SampledSoundList,
 SampleRate,Sound}
```

```
(*listen to it *)
sound = Import[fname,"Sound"];
EmitSound[sound]

fs = Import[fname,"SampleRate"]
```

```
22050
```

now load the samples and do power spectrum

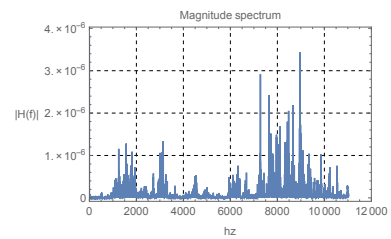
```
data = Import[fname,"Data"];
{nChannel,nSamples}=Dimensions[data]
```

```
Out[115]= {2,29016}
```

```
py = Fourier[data[[1,All]],
        FourierParameters->{1,-1}];
nUniquePts = Ceiling[(nSamples+1)/2]
```

```
Out[117]= 14509
```

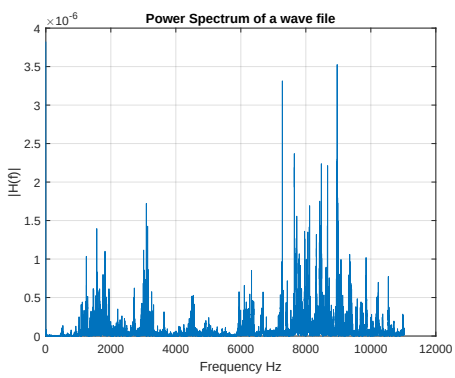
```
py = py[[1;;nUniquePts]];
py = Abs[py];
py = py/nSamples;
py = py^2;

If[ OddQ[nSamples],
  py[[2;;-1]]=2*py[[2;;-1]],
  py[[2;;-2]]=2*py[[2;;-2]]
];

f = (Range[0,nUniquePts-1] fs)/nSamples;
ListPlot[Transpose[{f,py}],
  Joined->True,
  FrameLabel->{{"|H(f)|",None},
               {"hz","Magnitude spectrum"}},
  ImageSize->400,
  Frame->True,
  RotateLabel->False,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  PlotRange->{{0,12000},All}]
```



3.10.2 Matlab

```
clear all; close all;
%
%This script plots the frequency spectrum of a wave file.
%This method of plotting the frequency spectrum is modeled after the
%example on Mathworks website tech note 1702

[data,Fs]=audioread('stereo1.wav');
[nSamples,nChannels]=size(data);
waveFileLength=nSamples/Fs;

N = 2^(nextpow2(length(data(:,1))));

Y=fft(data(:,1),N);
```

```
NumUniquePts = ceil((N+1)/2);
Y = Y(1:NumUniquePts);
P = abs(Y);
P = P/length(data(:,1));
P = P.^2;

if rem(N, 2) % odd nfft excludes Nyquist point
  P(2:end) = P(2:end)*2;
else
  P(2:end -1) = P(2:end -1)*2;
end

f = (0:NumUniquePts-1)*Fs/N;
plot(f,P);
title(sprintf('Power Spectrum of a wave file'));
xlabel('Frequency Hz');
ylabel('|H(f)|');
grid;
print(gcf, '-dpdf', '-r600', 'images/matlab_e52_1');
```



3.11 Plot the power spectral density of a signal

Problem: Given a signal

$$3\sin(2\pi f_1 t) + 4\cos(2\pi f_2 t) + 5\cos(2\pi f_3 t)$$

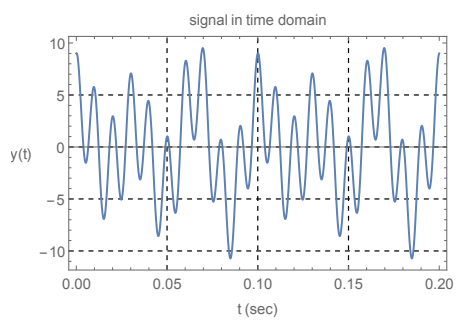
for the following frequency values (in Hz)

$$f_1 = 20, f_2 = 30, f_3 = 50$$

and plot the signal power spectral density so that 3 spikes will show at the above 3 frequencies. Use large enough sampling frequency to obtain a sharper spectrum

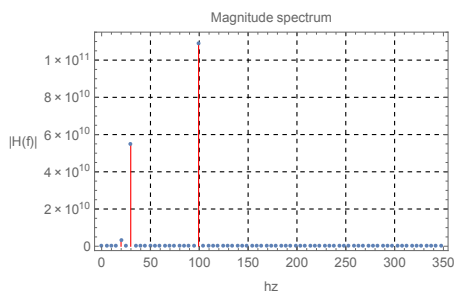
3.11.1 Mathematica

```
Clear ["Global`*"];
f1 = 20;
f2 = 30;
f3 = 100;
maxTime = 0.2;

y[t_]:=2 Sin[2 Pi f1 t]+4 Cos[2 Pi f2 t]+
       5 Cos[2 Pi f3 t];

Plot[y[t],{t,0,maxTime},
  Frame->True,
  FrameLabel->{{"y(t)",None},
      {"t (sec)","signal in time domain"}},
  RotateLabel->False,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  PlotRange->All,BaseStyle -> 12]
```

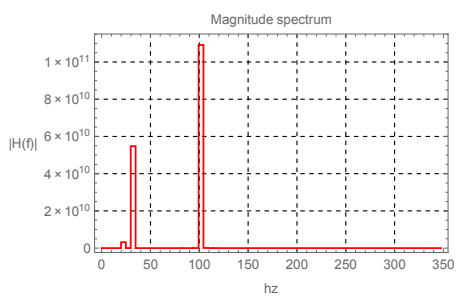



```
fs  = 7.0*Max[{f1,f2,f3}];
yn  = Table[y[n],{n,0,maxTime,(1/fs)}];
len = Length[yn];
py  = Fourier[yn];
nUniquePts = Ceiling[(len+1)/2];
py  = py[[1;;nUniquePts]];
py  = Abs[py];
py  = 2*(py^2);
```

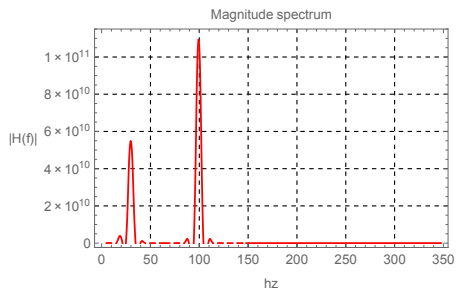
```
py[[1]] = py[[1]]/2;
f   = (Range[0,nUniquePts-1] fs)/len;

ListPlot[Transpose[{f,py}],Joined->False,
  FrameLabel->{{"|H(f)|",None},
               {"hz","Magnitude spectrum"}},
  ImageSize->400,
  Filling->Axis,
  FillingStyle->{Thick,Red},
  Frame->True,
  RotateLabel->False,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  PlotRange->All,BaseStyle -> 12]
```



Here is the same plot as above, but using `InterpolationOrder -> 0`

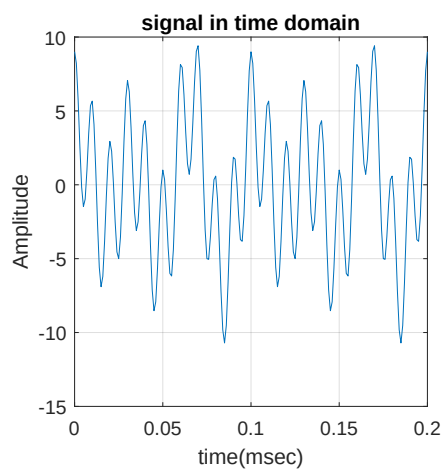


And using `InterpolationOrder -> 2`



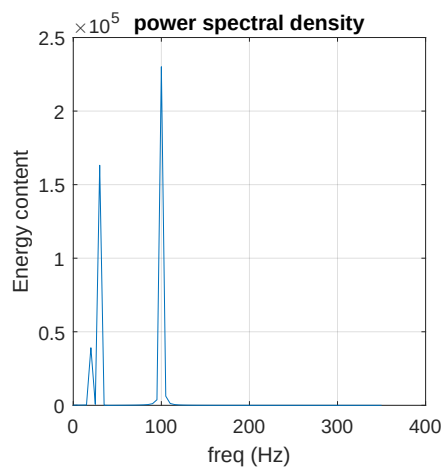
3.11.2 Matlab

```
clear all; close all;
maxTime = 0.2;
t  = 0:0.001:maxTime;
f1 =20; %Hz
f2 =30; %Hz
f3 =100; %Hz
y = @(t) 2*sin(2*pi*f1.*t)+4*cos(2*pi*f2.*t)+...
         5*cos(2*pi*f3.*t);
plot(t,y(t));
set(gcf,'Position',[10,10,320,320]);
grid on
title('signal in time domain');
xlabel('time(msec)');
ylabel('Amplitude');
```




```
fs  = 7.0*max([f1 f2 f3]);
```

```
delT=1/fs;
yn  = y(linspace(0,maxTime,maxTime/delT));
len = length(yn);
py  = fft(yn);
nUniquePts = ceil((len+1)/2);
py  = py(1:nUniquePts);
py  = abs(py);
py  = 2*(py.^2);
py(1) = py(1)/2;
f   = 0:nUniquePts-1;
f   = f*fs/len;
plot(f,py);
title('power spectral density');
xlabel('freq (Hz)');
ylabel('Energy content');
grid on
set(gcf,'Position',[10,10,320,320]);
```



3.12 Display spectrum of 2D image

3.12.1 Mathematica

```
img=Import["ExampleData/lena.tif"];
Image[img,ImageSize->300]
```

```
ImageDimensions[img]
```

```
{150,116}
```

```
(*see how many channels*)
ImageChannels[img]
```

```
3
```

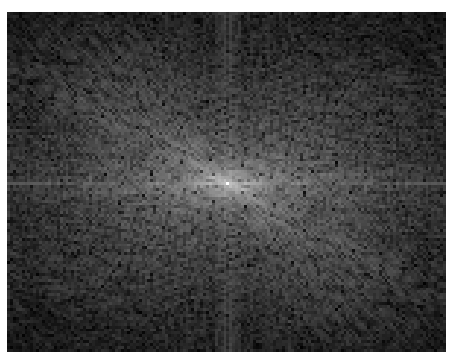
```
data=ImageData[img];  (*get data*)
{nRow,nCol,nChannel}=Dimensions[data]
```

```
{116,150,3}
```

```
(*look at each channel*)
```

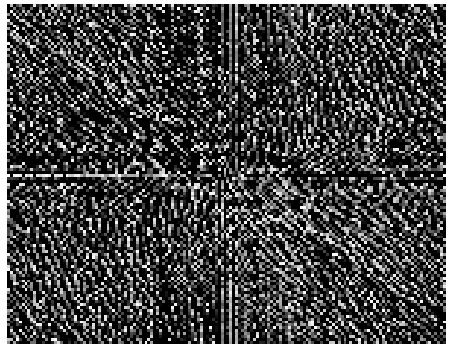
```
Map[Image[data[[All,All,#]],ImageSize->100]&,
        Range[1,nChannel]]
```

```
(*get channel 2 to FFT but center it first*)
d = data[[All,All,2]];
d = d*(-1)^Table[i+j,{i,nRow},{j,nCol}];

(*make FFT,center, view spectrum and phase*)
fw = Fourier[d,FourierParameters->{1,1}];
(*adjust for better viewing as needed*)
fudgeFactor = 100;
abs = fudgeFactor * Log[1+Abs@fw];
Labeled[Image[abs/Max[abs],
       ImageSize->300],
       Style["Magnitude spectrum", 18]]
```



Magnitude spectrum

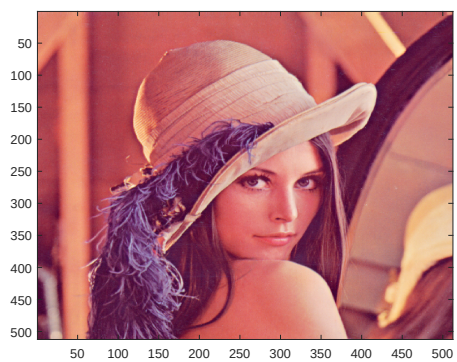
```
arg = Arg@fw;
Labeled[Image[arg/Max[arg],
        ImageSize->300],
        Style["Phase spectrum", 18]]
```



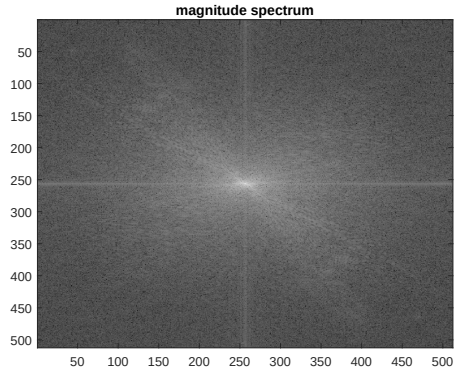
Phase spectrum

3.12.2 Matlab

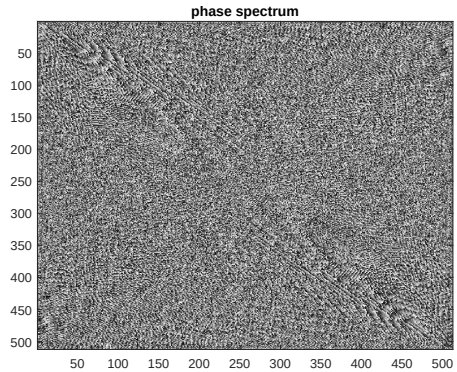
```
close all; clear all;
%image in same folder.
img = imread('lena.tiff','tif');
imagesc(img)
```




```
img = fftshift(img(:,:,2));
F = fft2(img);
figure;
imagesc(100*log(1+abs(fftshift(F))));
colormap(gray);
title('magnitude spectrum');
```




```
figure;
imagesc(angle(F));  colormap(gray);
title('phase spectrum');
```



3.13 Obtain the statistical maximum likelihood estimates (MLE) of probability distributions

This example uses the normal distribution and Poisson as examples. The maximum likelihood estimates of the population parameters is found by solving equation(s) using the standard method of taking logs and differentiation to solve for the maximum. Mathematica and Matlab version will be added at later time.

3.13.1 Maple

```
restart;   with(Statistics):
X := RandomVariable(Normal(mu,sigma));
lik:=product(PDF(X,x[i]),i=1..n);
lik:=expand(ln(lik)) assuming positive;
eq1:=diff(lik,mu)=0;
eq2:=diff(lik,sigma)=0;
solve({eq1,eq2},{mu,sigma});
allvalues( %[2]);
```

```
> restart;   with(Statistics):
> X := RandomVariable(Normal(mu,sigma));
```

$$X := \_R$$

```
> lik:=product(PDF(X,x[i]),i=1..n);
```

$$lik := \prod_{i=1}^{n} \frac{\sqrt{2}\, e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}}{2\sqrt{\pi}\,\sigma}$$

```
> lik:=expand(ln(lik)) assuming positive;
```

$$lik := \sum_{i=1}^{n} \ln\left(\frac{\sqrt{2}\, e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}}{2\sqrt{\pi}\,\sigma}\right)$$

```
> eq1:=diff(lik,mu)=0;
```

$$eq1 := -\frac{n\mu}{\sigma^2} + \left(\sum_{i=1}^{n} \frac{x_i}{\sigma^2}\right) = 0$$

```
> eq2:=diff(lik,sigma)=0;
```

$$eq2 := -\frac{n}{\sigma} + \frac{n\mu^2}{\sigma^3} + \sum_{i=1}^{n}\left(-\frac{2x_i\mu}{\sigma^3} + \frac{x_i^2}{\sigma^3}\right) = 0$$

```
> solve({eq1,eq2},{mu,sigma});
```

$$\left\{\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \sigma = 0\right\}, \left\{\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \sigma = \frac{RootOf\left(-\left(\sum_{i=1}^{n} x_i^2\right) n + \left(\sum_{i=1}^{n} x_i\right)^2 + \_Z^2\right)}{n}\right\}$$

```
> allvalues(%[2]);
```

$$\left\{\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \sigma = \frac{\sqrt{\left(\sum_{i=1}^{n} x_i^2\right) n - \left(\sum_{i=1}^{n} x_i\right)^2}}{n}\right\}, \left\{\mu = \frac{\sum_{i=1}^{n} x_i}{n}, \sigma = -\frac{\sqrt{\left(\sum_{i=1}^{n} x_i^2\right) n - \left(\sum_{i=1}^{n} x_i\right)^2}}{n}\right\}$$

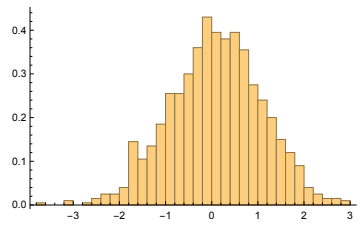
3.14 Make a histogram of data sampled from some probability distribution

This example uses the normal distribution. First random data is generated, then histogram of the data is made.

Matlab do not have an option, (unless I missed it) to make a relative histogram (this is where the total area is 1) but not hard to implement this.

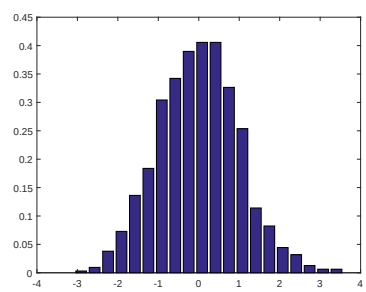
Mathematica

```
data = RandomReal[NormalDistribution[],1000];
Histogram[data,30,"ProbabilityDensity",
          ImageSize -> 300]]
```



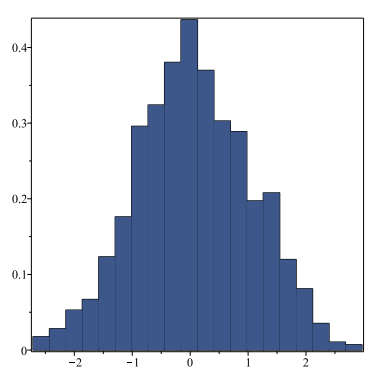
Matlab

```
data = randn(1000,1);
numberOfBins = 20;
[freq,bin]   = hist(data,numberOfBins);
binWidth     = (bin(end)-bin(1))/numberOfBins;
currentArea  = sum(binWidth*freq);
freq         = freq/currentArea;

bar(bin,freq)
```



Maple

```
restart;
with(Statistics):
nSamples:=1000:
data := Sample(RandomVariable(Normal(0, 1)),
                                  nSamples):

plots[display](Histogram(data,bincount=20,
                 frequencyscale=relative));
```



3.15 apply a filter on 1D numerical data (a vector)

Problem: suppose we want to apply say an averaging filter on a list of numbers. Say we want to replace each value in the list by the average 3 values around each value (a smoothing filter).

In Mathematica, ListConvolve is used, in Matlab conv() is used.

Mathematica

```
data={1,2,7,9,3,4,10,12};
filter=(1/3){1,1,1};
data[[2;;-2]]=ListConvolve[filter,data];
N[data]
```

```
Out[66]= {1.,
          3.33333,
          6.,
          6.33333,
          5.33333,
          5.66667,
          8.66667,
          12.}
```

Matlab

```
data   = [1 2 7 9 3 4 10 12];
filter = (1/3)*[1 1 1];
data(2:end-1) = conv(data,filter,'valid');
data'
```

```
    1.0000
    3.3333
    6.0000
    6.3333
    5.3333
    5.6667
    8.6667
   12.0000
```

3.16 apply an averaging Laplacian filter on 2D numerical data (a matrix)

Problem: Apply a Laplacian filter on 2D data. In Mathematica, ListConvolve is used, in Matlab conv2() is used.

Mathematica

```
data={{0,4,10,5,3},
      {4,4,1,8,5},
      {5,1,2,3,8},
      {8,6,8,8,10},
      {10,3,7,7,8}};

filter= (1/4){{0,1,0},
              {1,0,1},
              {0,1,0}};

data[[2;;-2,2;;-2]]=ListConvolve[filter,data];
N[%]
```

```
{{0., 4.,   10.,  5.,  3.},
{4., 2.5,  6.,   3.5, 5.},
{5., 4.25, 3.25, 6.5, 8.},
{8., 5.,   5.75, 7.,  10.},
{10.,3.,   7.,   7.,  8.}}
```

Matlab

```
data=[0 4 10 5 3;
      4,4,1,8,5;
      5,1,2,3,8;
      8,6,8,8,10;
      10,3,7,7,8];

filter= (1/4)*[0,1,0;
              1,0,1;
              0,1,0];

data(2:end-1,2:end-1)=...
     conv2(data,filter,'valid')
```

```
data =
  0       4.0000 10.0000  5.0000  3.0000
  4.0000  2.5000  6.0000  3.5000  5.0000
  5.0000  4.2500  3.2500  6.5000  8.0000
  8.0000  5.0000  5.7500  7.0000 10.0000
 10.0000  3.0000  7.0000  7.0000  8.0000
```

3.17 How to find cummulative sum

compute $\sum_{k=1}^{10} \frac{1}{k(k+1)}$

Mathematica

```
N[Sum[1/(k*(k + 1)), {k, 10}]]
```

```
0.9090909090909091
```

Matlab

```
format long
k=1:10;
s=cumsum(1./(k.*(k+1)));
s(end)
```

```
0.909090909090909
```

3.18 How to find the moving average of a 1D sequence?

Given some sequence such as $1, 2, 3, 4, 5, 6, 7$ how to find the moving average for different window sizes?

Mathematica

For window size $k = 2$

```
v = {1, 2, 3, 4, 5, 6, 7, 8};
f = {1/2, 1/2};
ListConvolve[f, v] // N
```

```
{1.5,2.5,3.5,4.5,5.5,6.5,7.5}
```

For window size $k = 3$

```
v = {1, 2, 3, 4, 5, 6, 7, 8};
f = Table[1/3, {3}];
ListConvolve[f, v] // N
```

```
{2., 3., 4., 5., 6., 7.}
```

Matlab

For a window size $k = 2$

```
V=[1 2 3 4 5 6 7 8];
f=[1/2 1/2];
conv(V,f,'valid')
```

```
ans =
 1.5000 2.5000 3.5000 4.5000 5.5000 6.5000 7.5000
```

For window size $k = 3$

```
V = [1 2 3 4 5 6 7 8];
k = 3;
f = ones(k,1)/k;
conv(V,f,'valid')
```

```
ans =
 2.0000 3.0000 4.0000 5.0000 6.0000 7.0000
```

3.19 How to select N random values from a set of numbers?

Given the set $v1, 2, 3, 5, 6, 7, 11, 12, 20, 21$ how to select say $m = 5$ random numbers from it?

Mathematica

method 1

```
a = {1, 2, 3, 5, 6, 7, 11, 12, 20, 21};
m = 5;
b = Table[0, {m}];
Do[k = RandomInteger[{1, Length[a]}];
  b[[i]] = a[[k]];
  a = Delete[a, k],
  {i, 1, m}
  ];
b
```

```
{6, 21, 3, 5, 11}
```

method 2 (Version 9)

```
RandomSample[a, m]
```

```
{1, 6, 11, 7, 20}  *)
```

Matlab

```
A = [1,2,3,5,6,7,11,12,20,21];
m = 5;
B = zeros(m,1);
for i = 1:m
    k = randi(length(A),1);
    B(i) = A(k);
    A(k) = [];
end
B
```

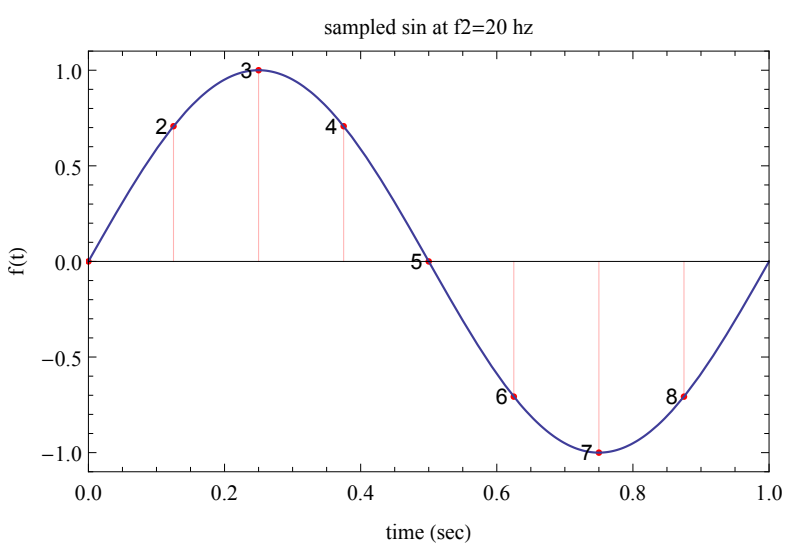
```
B =
     2
    20
     7
    11
     1
```

3.20 How to sample a sin signal and plot it?

Sample a sin signal of one second period at 8 times its frequency.

Mathematica

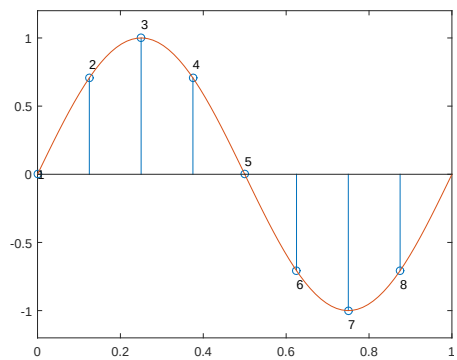
```
period   = 1; (*sec*)
f        = 1/period;
fs       = 8*f; (*sample at 8 times*)
Ts       = 1/fs; (*sampling period*)
nSamples = Round[period/Ts];
x        = Array[# &, nSamples, {0, period - Ts}]; (*linspace*)
signal   = {#, Sin[2 Pi f #]} & /@ x;
text     = MapIndexed[Text[ToString@First[#2], #1, {2, 0}] &, signal];

Show[
  ListPlot[signal, PlotStyle -> Red, Filling -> Axis, Frame -> True,
   FrameLabel -> {{"f(t)", None}, {"time (sec)", "sampled sin at f2=20 hz"}},
   Epilog -> text, PlotTheme -> "Classic",
   PlotRangePadding -> 0
   ],
  Plot[Sin[2 Pi f t], {t, 0, period}, PlotTheme -> "Classic"],
  BaseStyle -> 10, PlotRange -> {{0, period}, {-1.1, 1.1}}]
```



Matlab

```
clear all; close all;
```

```
period   = 1;
f        = 1/period;
fs       = 8*f;
Ts       = 1/fs;
nSamples = round(period/Ts);
x        = linspace(0,period-Ts,nSamples);
signal   = sin(2*pi*f*x);

stem(x,signal)
for i = 1:length(x)
    text(x(i),sign(signal(i))*0.1+signal(i),num2str(i))
end
hold on;
x  = linspace(0,period,10*nSamples);
plot(x,sin(2*pi*f*x))
ylim([-1.2,1.2]);
```



3.21 How find the impulse response of a difference equation?

Find the impulse response $h[n]$ for the discrete system given by the difference equation $y[n] - \frac{1}{2}y[n-1] = x[n] - \frac{1}{4}x[n-1]$

analytical solution:

1. The first step is to replace $y[n]$ by $h[n]$ and $x[n]$ by $\delta[n]$ giving

$$h[n] - \frac{1}{2}h[n-1] = \delta[n] - \frac{1}{4}\delta[n-1]$$

2. Taking the Fourier transform and assuming $H(e^{i\omega})$ is the Fourier transform of

$h[n]$ results in

$$H(e^{i\omega}) - \frac{1}{2}H(e^{i\omega})e^{-i\omega} = 1 - \frac{1}{4}e^{-i\omega}$$

3. Solving for $H(e^{i\omega})$ gives

$$\begin{aligned} H(e^{i\omega}) &= \frac{1 - \frac{1}{4}e^{-i\omega}}{1 - \frac{1}{2}e^{-i\omega}} \\ &= \frac{1}{1 - \frac{1}{2}e^{-i\omega}} - \frac{1}{4}\frac{e^{-i\omega}}{1 - \frac{1}{2}e^{-i\omega}} \end{aligned}$$

4. Taking the inverse discrete Fourier transform given by

$$h[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(e^{i\omega})e^{i\omega}$$

which results in

$$h[n] = \left(\frac{1}{2}\right)^n u[n] - \frac{1}{4}\left(\frac{1}{2}\right)^{n-1} u[n-1]$$

Mathematica

```
Clear[w, n]
expr1 = -(1/4)*Exp[-I w]/(1 - (1/2)*Exp[-I w]);
expr2 = 1/(1 - (1/2)*Exp[-I w]);
sol1 = InverseFourierSequenceTransform[expr1, w, n];
sol2 = InverseFourierSequenceTransform[expr2, w, n];
sol = sol1 + sol2
```

$$\begin{matrix} -2^{-n-1} & n > 0 \\ 0 & \text{True} \end{matrix} + \begin{matrix} 2^{-n} & n \geq 0 \\ 0 & \text{True} \end{matrix}$$

And some values of $h[n]$ starting from $n = 0$ are

```
(sol /. n -> #) & /@ Range[0, 10]
```

$$\left\{1, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{512}, \frac{1}{1024}, \frac{1}{2048}\right\}$$

CHAPTER 4

Differential, PDE solving, integration, numerical and analytical solving of equations

4.1 Generate direction field plot of a first order differential equation

Problem: Given the following non autonomous differential equation, plot the line fields which represents the solutions of the ODE.

$$\frac{dy(x)}{dx} = x^2 - y$$

Direction field plot (or slope plot) shows solutions for the ODE without actually solving the ODE.

The following are the steps to generate direction field plot for $\frac{dy}{dx} = f(x, y)$

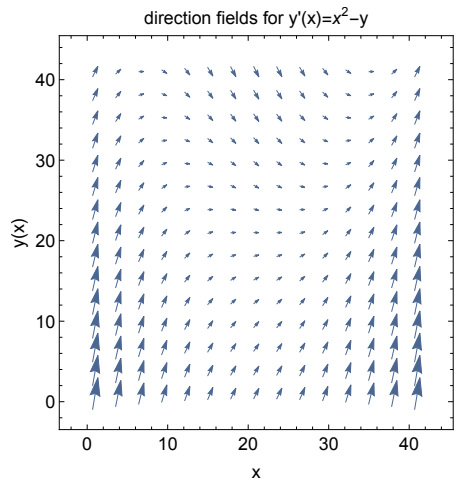
1. generate 2 lists of numbers. The y list and the x list. These 2 lists are the coordinates at which a small slop line will be plotted.
2. At each of the above coordinates (y, x) evaluate $f(x, y)$.
3. Plot small line starting at the point (x, y) and having slop of $f(x, y)$. The length of the line is kept small. Normally it has an arrow at the end of it.

Using Matlab, the points are first generated (the (x, y) coordinates) then the slope $f(x, y)$ evaluated at each of these points, then the command quiver() is used. Next contour() command is used to plot few lines of constant slope.

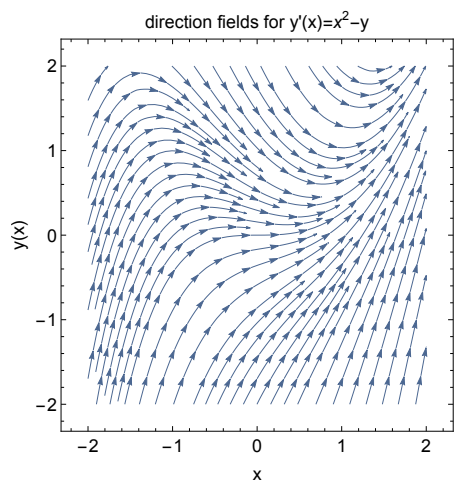
In Mathematica, the command `VectorPlot` is used. In Maple `dfieldplot` is used.

Mathematica

```
f[x_,y_]:= x^2-y
ListVectorPlot[Table[{1,f[x,y]},
        {x,-2,2,0.1},{y,-2,2,0.1}],
      FrameLabel->{{"y(x)",None},
     {"x",
      "direction fields for y'(x)=x^2-y"}
     },
      ImageSize->350,
      LabelStyle->Directive[14]]
```

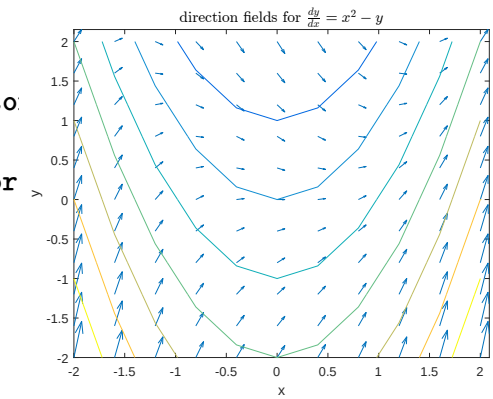



```
StreamPlot[{1,f[x,y]},{x,-2,2},{y,-2,2},
 FrameLabel->{{"y(x)",None},
  {"x","direction fields for y'(x)=x^2-y"}
 ImageSize->350,
 LabelStyle->Directive[14]]
```



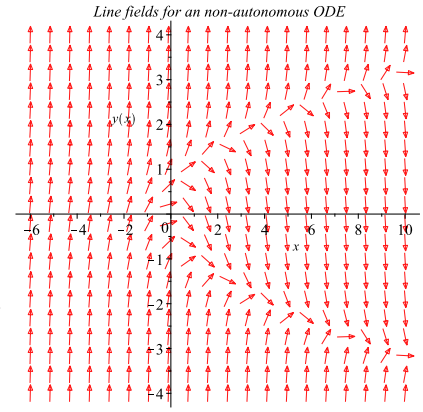
Matlab

```
clear all; close all;

%direction fields for dy/dx=xy
f     = @(X,Y) X.^2 - Y;
[X,Y] = meshgrid(-2:.4:2,-2:.4:2);
YlengthOfVector = f(X,Y);
XlengthOfVector = ones(size(YlengthOfVecto

quiver(X,Y,XlengthOfVector,YlengthOfVector
xlabel('x'); ylabel('y');
hold on;

contour(X,Y,YlengthOfVector,...
  [-5 -4 -3 -2 -1 0 1 2 3 4 5]);

title(...
'direction fields for $\frac{dy}{dx}=x^2-y$',...
'interpreter','latex','fontsize',12)
```



Maple

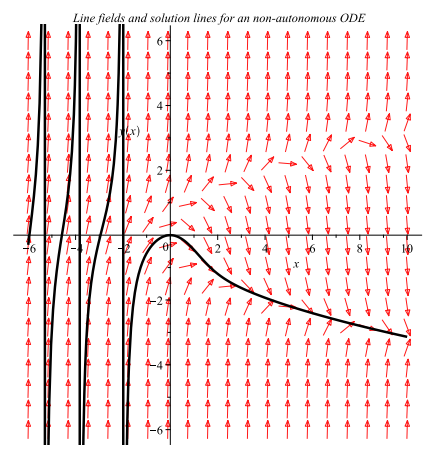
```
restart;
with(DEtools): with(plots):
ode1:=diff(y(x),x)=(y(x))^2-x;
dfieldplot(ode1,y(x),x=-6..10, y=-4..4,
 arrows=MEDIUM,
 title=`Line fields for an non-autonomous ODE`
```




```
restart;
with(DEtools): with(plots):
ode1:=diff(y(x),x)=(y(x))^2-x:
p1:= dfieldplot(ode1,y(x),x=-6..10, y=-6..6,
       arrows=MEDIUM):
sol:=dsolve({ode1,y(0)=0});
```

$$y(x) = -\frac{\sqrt{3}\mathrm{Ai}^{(1)}(x) + \mathrm{Bi}^{(1)}(x)}{\sqrt{3}\mathrm{Ai}\,(x) + \mathrm{Bi}\,(x)}$$

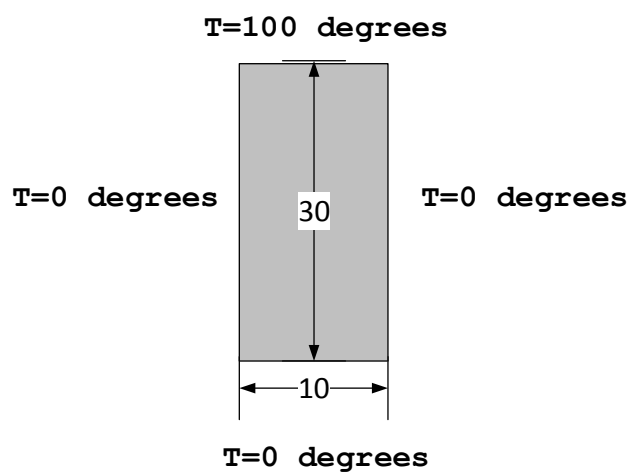
```
p2:=plot(rhs(sol),x=-6..10,
       color=black,thickness=3):
display({p1,p2},
 title=`Line fields and solution lines
     for an non-autonomous ODE`);
```



4.2 Solve the 2-D Laplace PDE for a rectangular plate with Dirichlet boundary conditions

Problem: Solve $\nabla^2 T(x,y) = 0$ on the following plate, with height $h = 30$, and width $w = 10$, and with its edges held at fixed temperature as shown, find the steady state temperature distribution

System boundary conditions.



Plate, with sides held at the tempretures shown. Find T(x,y), the heat distribution anywhere inside the plate.

Mathematica `NDSolve[]` does not currently support Laplace PDE as it is not an initial value problem.

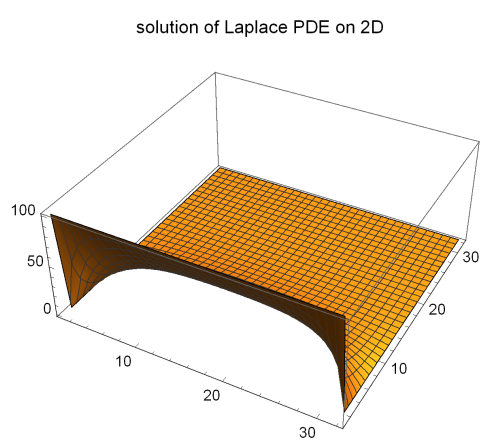
Jacobi iterative method is used below to solve it. 100 iterations are made and then the resulting solution plotted in 3D.

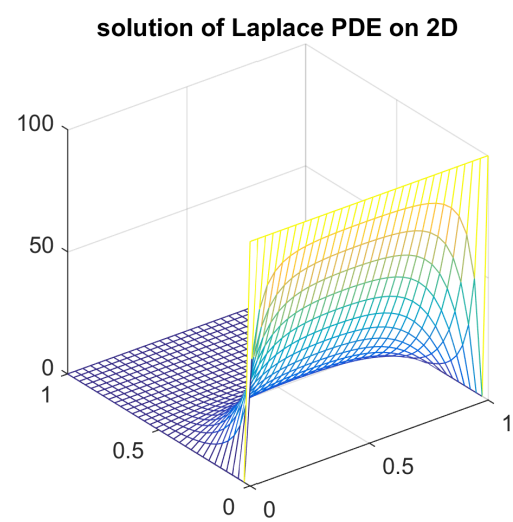
Mathematica

```
n = 32;  (*grid size*)
h = 1/(n-1);(*grid spacing*)
u = Table[0.,{n},{n}]; (*init to zero
u[[1,All]] = 100; (*B.C.*)

Do[ (*iterate 100 times*)
  tmp=u;
  For[i=2,i<=n-1,i++,
    For[j=2,j<=n-1,j++,
      tmp[[i,j]]=
        (1/4)(u[[i-1,j]]+u[[i+1,j]]+
           u[[i,j-1]]+u[[i,j+1]])
    ]
  ];

  u=tmp,
  {100}
];

ListPlot3D[u,PlotRange->All,
 ImagePadding->20,
 Mesh->{n,n},
 ImageSize->300,
 PlotLabel->"solution of Laplace PDE o
]
```

Matlab

```
n = 32;
h = 1/(n-1); %grid spacing
u = zeros(n,n);
u(1,:) = 100; %B.C.

% coordinates
[X,Y]  = meshgrid(0:h:1,0:h:1);

for k=1:100
    tmp = u;

    for i=2:n-1
        for j=2:n-1
          tmp(i,j)=...
           (1/4)*(u(i-1,j)+u(i+1,j)+...
             u(i,j-1)+u(i,j+1));
        end
    end

    u=tmp;
end

mesh(X,Y,u);
title('solution of Laplace PDE on 2D');
set(gcf,'Position',[10,10,320,320]);
```





4.3 Solve homogeneous 1st order ODE, constant coefficients and initial conditions

Problem: Solve

$$y'(t) = 3y(t)$$

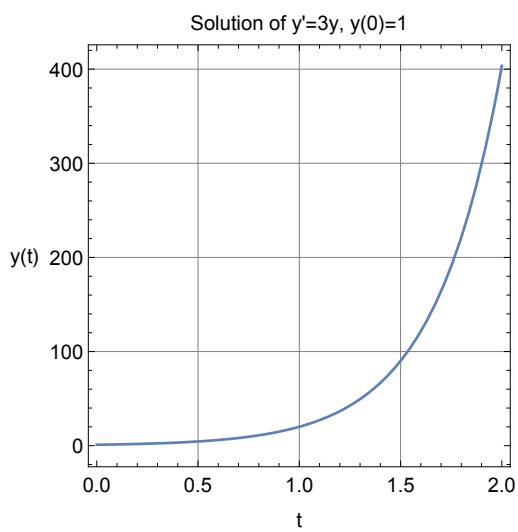
with initial condition $y(0) = 1$ and plot the solution for $t = 0 \cdots 2$ seconds.

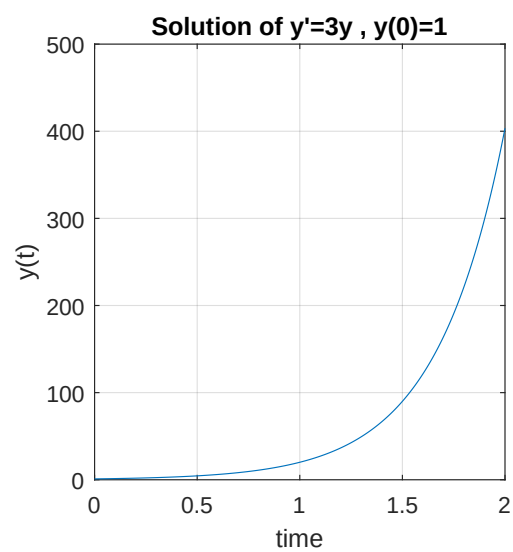
Mathematica

```
Remove["Global`*"];
sol = First@DSolve[{y'[t]==3y[t],
         y[0]==1},y[t],t];
y   = y[t]/.sol

       Out[26]= E^(3 t)

Plot[y,{t,0,2},
  FrameLabel->{{"y(t)",None},
   {"t","Solution of y'=3y, y(0)=1"}},
  Frame->True,
  GridLines->Automatic,
  GridLinesStyle->Automatic,
  RotateLabel->False,
  ImageSize->300,
  AspectRatio->1]
```

Matlab

```
function e53
t          = 0:0.001:2;   % time
initial_y = 1;

[t,y] = ode45( @rhs, t, initial_y);

plot(t,y)
title('Solution of y''=3y , y(0)=1');
xlabel('time');
ylabel('y(t)');
grid on
set(gcf,'Position',[10,10,420,420]);

    function dydt=rhs(t,y)
        dydt = 3*y;
    end
end
```





4.4 Solve homogeneous 2nd order ODE with constant coefficients

Problem: Solve

$$y''(t) - 1.5y'(t) + 5y(t) = 0$$

with initial conditions

$$y(0) = 1, y'(0) = 0$$

To use Matlab ode45, the second order ODE is first converted to state space formulation as follows

Given $y''(t) - 1.5y'(t) + 5y(t) = 0$ let

$$\begin{aligned} x_1 &= y \\ x_2 &= y' \\ &= x_1' \end{aligned}$$

hence

$$x_1' = x_2$$

and

$$\begin{aligned} x_2' &= y'' \\ &= 1.5y' - 5y \\ &= 1.5x_2 - 5x_1 \end{aligned}$$

Hence we can now write

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

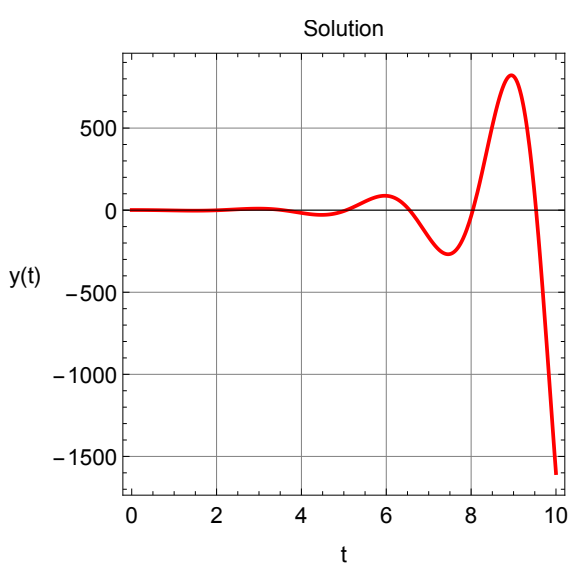
Now Matlab ODE45 can be used.

Mathematica

```
Remove["Global`*"];
eq  = y''[t]-1.5y'[t]+5y[t]==0;
ic  = {y'[0]==0,y[0]== 1};
sol = First@DSolve[{eq,ic},y[t],t];
y   = y[t]/.sol
```

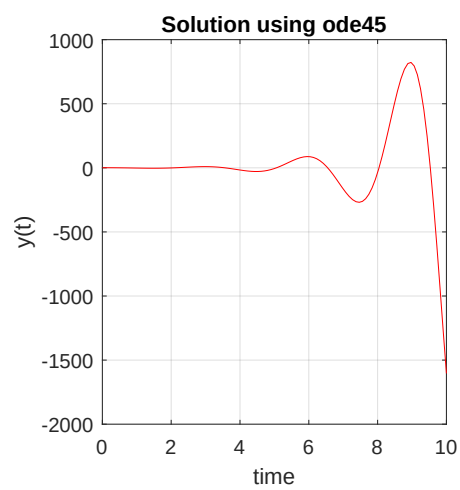
$1.\left(1.e^{0.75t}\cos(2.10654t) - 0.356034e^{0.75t}\sin(2.1065\right.$

```
Plot[y,{t,0,10},
     FrameLabel->{{"y(t)",None},
         {"t","Solution"}},
     Frame->True,
     GridLines->Automatic,
     GridLinesStyle->Automatic,
     RotateLabel->False,
     ImageSize->300,
     AspectRatio->1,
     PlotRange->All,
     PlotStyle->{Thick,Red}]
```



Matlab

```
function e54

t0 = 0; %initial time
tf = 10; %final time

%initial conditions [y(0)  y'(0)]
ic =[1 0]';

[t,y] = ode45(@rhs, [t0 tf], ic);

plot(t,y(:,1),'r')
title('Solution using ode45');
xlabel('time');
ylabel('y(t)');
grid on
set(gcf,'Position',[10,10,320,320]);

    function dydt=rhs(t,y)
        dydt=[y(2) ;
              -5*y(1)+1.5*y(2)];
    end
end
```



4.5 Solve non-homogeneous 2nd order ODE, constant coefficients

Problem: Solve

$$y''(t) - 1.5y'(t) + 5y(t) = 4\cos(t)$$

with initial conditions

$$y(0) = 1, y'(0) = 0$$

To use Matlab ode45, the second order ODE is converted to state space as follows

Given $y''(t) - 1.5y'(t) + 5y(t) = 4\cos(t)$, let

$$\begin{aligned} x_1 &= y \\ x_2 &= y' \\ &= x_1' \end{aligned}$$

hence

$$x_1' = x_2$$

and

$$\begin{aligned} x_2' &= y'' \\ &= 1.5y' - 5y + 4\cos(t) \\ &= 1.5x_2 - 5x_1 + 4\cos(t) \end{aligned}$$

Hence we can now write

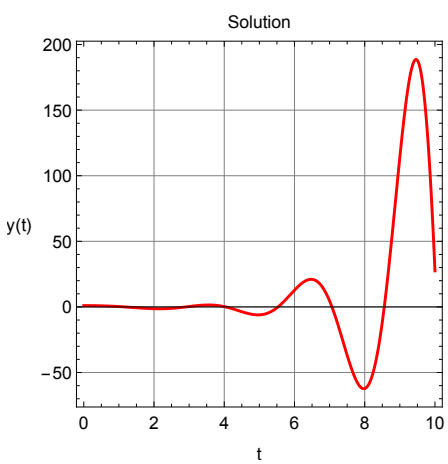
$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -5 & 1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 4 \end{bmatrix} \cos(t)$$

<u>Mathematica</u>

```
Remove["Global`*"];
eq  = y''[t]-15/10y'[t]+5y[t]==4  Cos[t];
ic  = {y'[0]==0,y[0]== 1};
sol = First@DSolve[{eq,ic},y[t],t];
```

$$\frac{1}{5183}(-1704\sin(t)\sin^2\left(\frac{\sqrt{71}t}{4}\right)$$
$$+69\sqrt{71}e^{3t/4}\sin\left(\frac{\sqrt{71}t}{4}\right)+4544\cos(t)\cos^2\left(\frac{\sqrt{71}t}{4}\right)$$
$$+639e^{3t/4}\cos\left(\frac{\sqrt{71}t}{4}\right)-1704\sin(t)\cos^2\left(\frac{\sqrt{71}t}{4}\right)+4544\sin^2\left(\frac{\sqrt{71}t}{4}\right)\cos(t)))$$

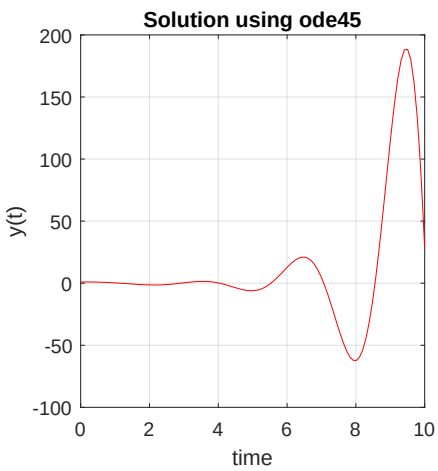
```
Plot[y,{t,0,10},
     FrameLabel->{{"y(t)",None},
      {"t","Solution"}},
     Frame->True,
     GridLines->Automatic,
     GridLinesStyle->Automatic,
     RotateLabel->False,
     ImageSize->300,
     AspectRatio->1,
     PlotRange->All,
     PlotStyle->{Thick,Red}]
```



Matlab

```
function e55
t0 = 0; %initial time
tf = 10; %final time
 %initial conditions [y(0)  y'(0)]
ic =[1 0]';
```

```
[t,y] = ode45(@rhs, [t0 tf], ic);

plot(t,y(:,1),'r')
title('Solution using ode45');
xlabel('time');
ylabel('y(t)');
grid on
set(gcf,'Position',[10,10,320,320]);

  function dydt=rhs(t,y)
    dydt=[y(2) ;
         -5*y(1)+1.5*y(2)+4*cos(t)];
  end
end
```



4.6 Solve homogeneous 2nd order ODE, constant coefficients (BVP)

Problem: Solve

$$y''(t) + t\ y(t) = 0$$

with the boundary conditions

$$y(0) = 3, y(20) = -1$$

For solving with Matlab, the ODE is first converted to state space as follows

Given $y''(t) + t\, y(t) = 0$, let

$$\begin{aligned} x_1 &= y \\ x_2 &= y' \\ &= x_1' \end{aligned}$$

Therefore

$$x_1' = x_2$$

And

$$\begin{aligned} x_2' &= y'' \\ &= -t\, y \\ &= -t\, x_1 \end{aligned}$$

This results in

$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -t & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Now bvp4c() can be used.

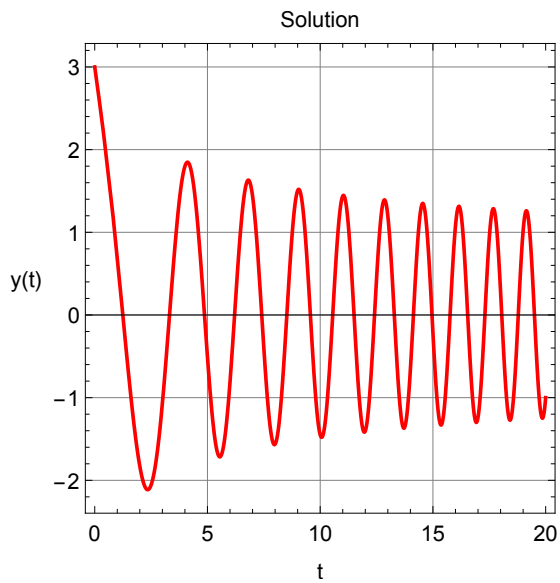
Mathematica

```
Clear[y,t];
eq  = y''[t]+t y[t]==0;
ic  = {y[0]==3,y[20]==-1};
sol = First@DSolve[{eq,ic},y[t],t];

y   = y[t]/.sol
```

$$\frac{-\sqrt{3}\mathrm{Ai}\left(\sqrt[3]{-1}t\right) + \mathrm{Bi}\left(\sqrt[3]{-1}t\right) - 3\ 3^{2/3}\Gamma\left(\frac{2}{3}\right)\mathrm{Bi}\left(20\sqrt[3]{-1}\right)\mathrm{Ai}\left(\sqrt[3]{-1}t\right) + 3\ 3^{2/3}\Gamma\left(\frac{2}{3}\right)\mathrm{Ai}\left(20\sqrt[3]{-1}\right)\mathrm{Bi}\left(\sqrt[3]{-1}t\right)}{\sqrt{3}\mathrm{Ai}\left(20\sqrt[3]{-1}\right) - \mathrm{Bi}\left(20\sqrt[3]{-1}\right)}$$

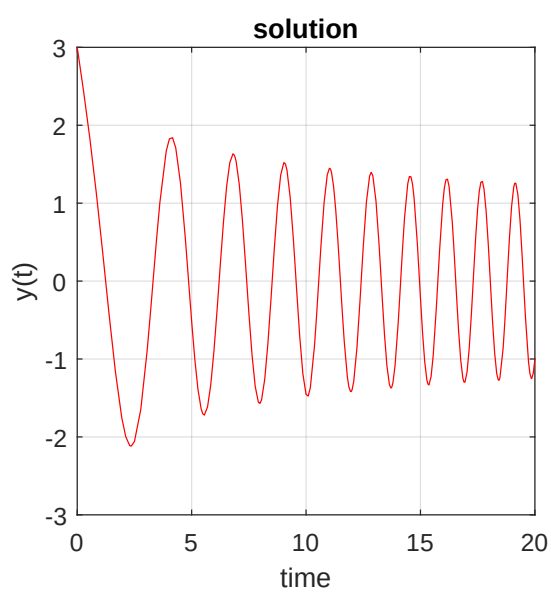
```
Plot[Re[y],{t,0,20},
     FrameLabel->{{"y(t)",None},
      {"t","Solution"}},
     Frame->True,
     GridLines->Automatic,
     GridLinesStyle->Automatic,
     RotateLabel->False,
     ImageSize->300,
```

```
    AspectRatio->1,
    PlotRange->All,
    PlotStyle->{Thick,Red},
    Exclusions->None]
```



Matlab

```
function e56

t0 = 0; %initial time
tf = 20; %final time

solinit = bvpinit(linspace(t0,tf,5),[3 -1]);

sol = bvp4c(@odefun,@bcfun,solinit);

plot(sol.x(1,:),sol.y(1,:),'r')
title('solution');
xlabel('time');
ylabel('y(t)');
grid;
set(gcf,'Position',[10,10,320,320]);

function dydt=odefun(t,y)
dydt=[y(2); -t.*y(1)];

function res=bcfun(yleft,yright)
res=[yleft(1)-3
    yright(1)+1];
```



4.7 Solve the 1-D heat partial differential equation (PDE)

The PDE is

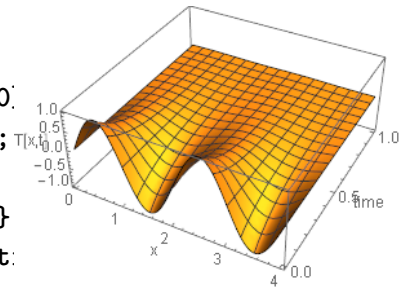
$$\frac{\partial T(x,t)}{\partial t} = k\frac{\partial^2 T(x,t)}{\partial x^2}$$

Problem: given a bar of length L and initial conditions $T(x,0) = \sin(\pi x)$ and boundary conditions $T(0,t) = 0, T(L,t) = 0$, solve the above PDE and plot the solution on 3D.

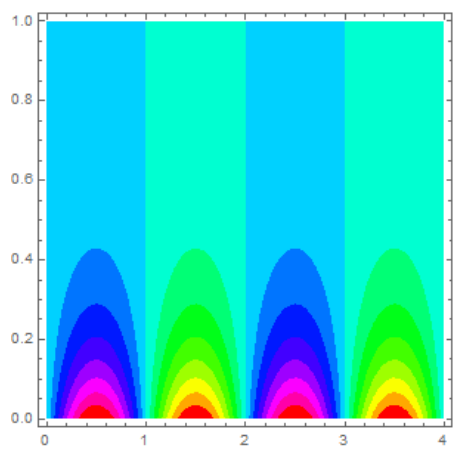
Use bar length of 4 and $k = 0.5$ and show the solution for 1 second.

Mathematica

```
Clear[y,x,t];
barLength = 4;
timeDuration = 1;
eq = D[y[x, t], t] == k*D[y[x, t], x, x];
eq = eq /. k -> 0.5;
boundaryConditions = {y[0,t]==0,
                      y[barLength,t]==0
initialConditions = y[x,0]==Sin[Pi x];
sol=First@NDSolve[{eq,
 boundaryConditions,initialConditions}
 y[x,t],{x,0,barLength},{t,0,timeDurat
y = y[x,t]/.sol
Plot3D[y,{x,0,barLength},{t,0,timeDuration},
  PlotPoints->30,PlotRange->All,
  AxesLabel->{"x","time","T[x,t]"},
  ImageSize->300]
```

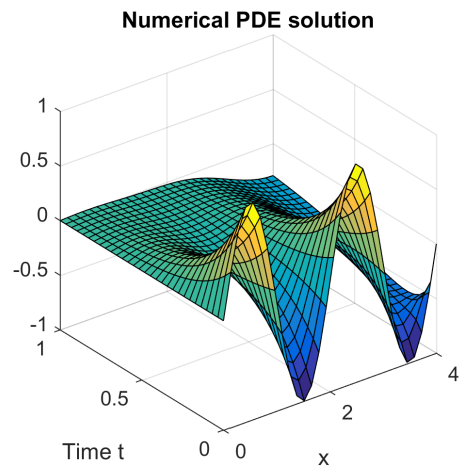



```
ContourPlot[y,{x,0,barLength},
  {t,0,timeDuration},PlotPoints->15,
  Contours->15,ContourLines->False,
  ColorFunction->Hue,PlotRange->All,
  ImageSize->300]
```



Matlab

```
function e57
m = 0;
x = linspace(0,4,30);
t = linspace(0,1,30);

sol = pdepe(m,@pde,
        @pdeInitialConditions,...
        @pdeBoundaryConditions,x,t);

u = sol(:,:,1);

surf(x,t,u)
title('Numerical PDE solution')
xlabel('x'); ylabel('Time t')
set(gcf,'Position',[10,10,320,320]);
% ----------------
function [c,f,s] = pde(x,t,u,DuDx)
k=0.5;
c = 1/k;
f = DuDx;
s = 0;
% ---------------
function T0 = pdeInitialConditions(x)
T0 = sin(pi*x);
% ---------------
function [pl,ql,pr,qr] = ...
   pdeBoundaryConditions(xl,ul,xr,ur,t)
pl = ul;
ql = 0;
pr = 0;
qr = 1;
```



4.8 Show the effect of boundary/initial conditions on 1-D heat PDE

The PDE is

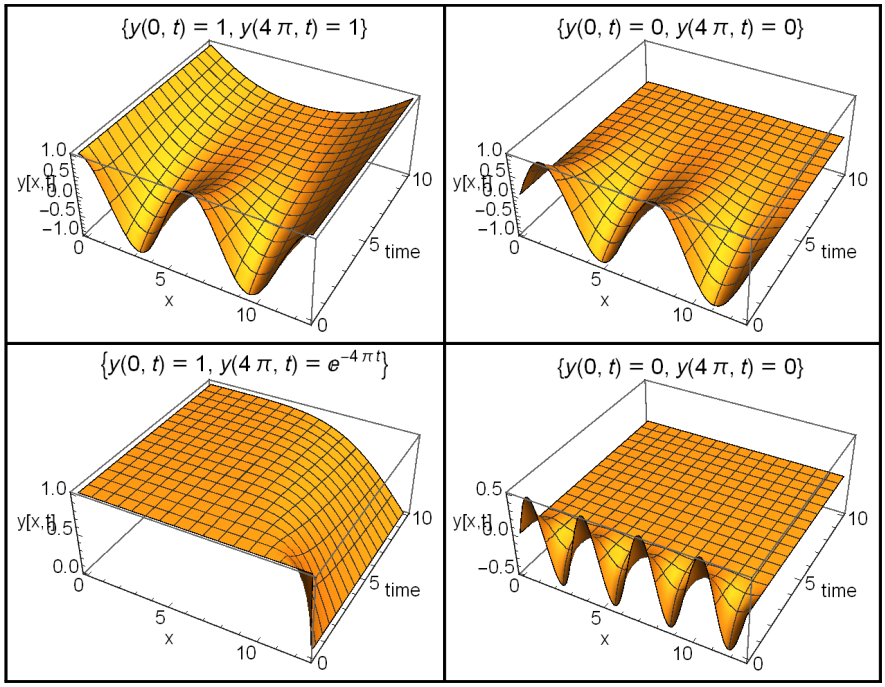
$$\frac{\partial T(x,t)}{\partial t} = k\frac{\partial^2 T(x,t)}{\partial x^2}$$

Problem: given a bar of length L , solve the above 1-D heat PDE for 4 different boundary/initial condition to show that the solution depends on these.

Mathematica

```
Clear[y,x,t,k];
SetDirectory[NotebookDirectory[]];
barLength = 4*Pi;
timeDuration = 10;
eq = D[y[x, t], t] == k*D[y[x, t], x, x];
eq = eq /. k -> 0.5;
solveHeat[eq_,bc_,ic_,y_,x_,t_,len_,timeDuration_]:=Module[{sol},
sol=First@NDSolve[{eq,bc,ic},y[x,t],{x,0,len},{t,0,timeDuration}];
Plot3D[y[x,t]/.sol,{x,0,barLength},{t,0,timeDuration},
       PlotPoints->30,PlotRange->All,AxesLabel->{"x","time","y[x,t]"},
       ImageSize->200,PlotLabel->bc]
];

bc={{y[0,t]==1,y[barLength,t]==1},
    {y[0,t]==0,y[barLength,t]==0},
    {y[0,t]==1,y[barLength,t]==Exp[-t barLength]},
    {y[0,t]==0,y[barLength,t]==0}
};

ic={y[x,0]== Cos[x],
y[x,0]==Sin[x],
y[x,0]==1,
y[x,0]== Cos[x]Sin[x]
};

sol = MapThread[solveHeat[eq,#1,#2,y,x,t,barLength ,timeDuration]&,{bc,ic}];
Grid[Partition[sol,2],Frame->All]
Export["images/mma_e58_1.pdf",%]
```

Each plot shows the boundary conditions used.



4.9 Find particular and homogenous solution to undetermined system of equations

Problem: Find the general solution to $Ax = b$

$$\begin{bmatrix} 2 & 4 & 6 & 4 \\ 2 & 5 & 7 & 6 \\ 2 & 3 & 5 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad where \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

In Maple 11, the LinearAlgebra package was used. In Mathematica one can also get the general solution, but one must find the Null space specifically and add it to the result from LinearSolve[] since LinearSolve[] finds particular solutions only.

In Matlab the same thing needs to be done. I am not sure now how to make Matlab give me the same particular solution as Maple and Mathematica since Matlab A\b uses least square approach to determine a solution. I am sure there is a way, will update this once I find out.

Mathematica

```
vars = {x1,x2,x3,x4};
eqs  ={2 x1+4 x2+6 x3+4 x4==4,
       2 x1+5 x2+7 x3+6 x4==3,
       2 x1+3 x2+5 x3+2 x4==5};

{b,mat} = CoefficientArrays[eqs,vars];
Normal[mat]
```

$$\begin{pmatrix} 2 & 4 & 6 & 4 \\ 2 & 5 & 7 & 6 \\ 2 & 3 & 5 & 2 \end{pmatrix}$$

```
Normal[b]
```

```
Out[163]= {-4,-3,-5}
```

Mathematica LinearSolve gives one solution (particular solution)

```
particularSolution = LinearSolve[mat,b]
```

```
Out[164]= {-4,1,0,0}
```

find the homogenous solution (nullspace) and add it to the above particular solution

```
homogenousSolution =Transpose[NullSpace[mat]]
```

$$\begin{pmatrix} 2 & -1 \\ -2 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Add the particular solution to the homogenous solution to get the general solution

```
fullSolution = particularSolution+homogenousSolution
```

$$\begin{pmatrix} -2 & -5 \\ -1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

To obtain the general solution right away Solve can be used instead

```
sol = Flatten[Solve[eqs,vars]]

During evaluation of
In[167]:= Solve::svars: Equations may
not give solutions for all
                 "solve" variables. >>
```

$$\left\{x3 \to -\frac{x1}{2} - \frac{x2}{2} + \frac{3}{2}, x4 \to \frac{x1}{4} - \frac{x2}{4} - \frac{5}{4}\right\}$$

Matlab

```
clear all;
A=[2 4 6 4;
   2 5 7 6;
   2 3 5 2]

b=[4 3 5]'
particularSolution=A\b
```

```
Warning: Rank deficient,
  rank = 2, tol =  4.033641e-15.

particularSolution =

         0
         0
    1.5000
   -1.2500
```

```
nullSolution=null(A,'r')
```

```
nullSolution =

    -1     2
    -1    -2
     1     0
     0     1
```

Maple

```
restart;
with(LinearAlgebra);
vars := x[1], x[2], x[3], x[4];
sys := 2*x[1]+4*x[2]+6*x[3]+4*x[4] = 4,
       2*x[1]+5*x[2]+7*x[3]+6*x[4] = 3,
       2*x[1]+3*x[2]+5*x[3]+2*x[4] = 5;

`eqs=`, convert([sys], Vector);
A, b := GenerateMatrix([sys], [vars])
```

$$A = \begin{bmatrix} 2 & 4 & 6 & 4 \\ 2 & 5 & 7 & 6 \\ 2 & 3 & 5 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

```
`general solution=`,
  LinearSolve(A, b, free = 'x')
```

$$\begin{bmatrix} 4 - x_3 + 2\,x_4 \\ -1 - x_3 - 2\,x_4 \\ x_3 \\ x_4 \end{bmatrix}$$

Can solve this system to get the general solution using the solve command as follows

```
s := solve([sys], [vars]);
r := subs(op(s), [vars]);
`general solution=`, convert(%, Vector)
```

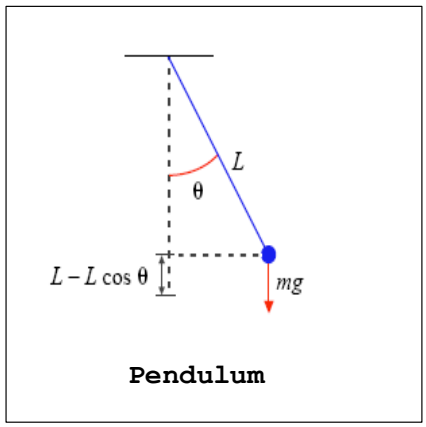
$$\begin{bmatrix} 4 - x_3 + 2\,x_4 \\ -1 - x_3 - 2\,x_4 \\ x_3 \\ x_4 \end{bmatrix}$$

WARNING: Maple sometimes reorders the result from solve() so we can get a different ordering of the free variables as shown above.

4.10 Plot the constant energy levels for a nonlinear pendulum

Problem:

Plot the constant energy levels for the nonlinear pendulum in $\theta, \dot{\theta}$



Assume that $m = 1, g = 9.8m/s^2, L = 10m$

Answer:

The constant energy curves are curves in the Y-X plane where energy is constant. The Y-axis represents $\dot{\theta}$, and the X-axis represents θ

We assume the pendulum is given an initial force when in the initial position ($\theta = 0^0$) that will cause it to swing anticlock wise. The pendulum will from this point obtain an energy which will remain constant since there is no damping.

The higher the energy the pendulum posses, the larger the angle θ it will swing by will be.

If the energy is large enough to cause the pendulum to reach $\theta = \pi$ (the upright position) with non zero angular velocity, then the pendulum will continue to rotate in the same direction and will not swing back and forth.

The expression for the energy E for the pendulum is first derived as a function of $\theta, \dot{\theta}$

$$\begin{aligned} E &= PE + KE \\ &= mgL(1 - \cos\theta) + \frac{1}{2}mL^2\dot{\theta}^2 \end{aligned}$$

The school PDF report contains more information on this topic.

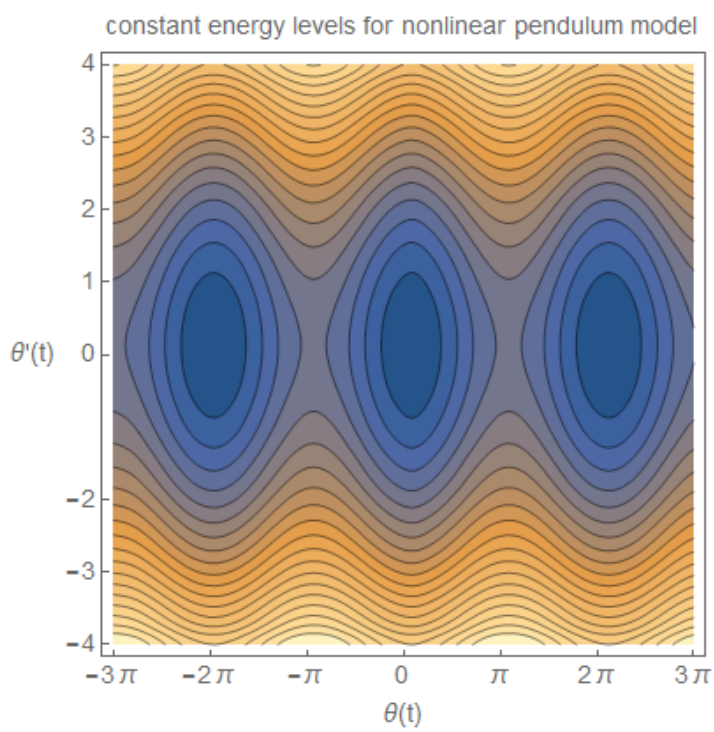
This was solved in Mathematica using the ListContourPlot[] command after generating the energy values.

The original Matlab implementation is left below as well as the Maple implementation. However, these were not done using the contour method, which is a much simpler method. These will be updated later.

The following is the resulting plot for $m = 1, g = 9.8m/s^2, L = 10m$

Mathematica

```
SetDirectory[NotebookDirectory[]];
energy[theta_, speed_, m_, g_, len_] := m*g*len*(1 - Cos[theta]) + (1/2)*m*len^2*speed^2;
m = 1; g = 9.8; len = 10;
data = Table[energy[i, j, m, g, len], {j, -4, 4, 0.01}, {i, -3*Pi, 3*Pi, Pi/20.}];
ListContourPlot[data, InterpolationOrder -> 2, Contours -> 20, MaxPlotPoints -> 30,
  DataRange -> {{-3*Pi, 3*Pi}, {-4, 4}},
  FrameTicks -> {{{-4, -3, -2, 0, 1, 2, 3, 4}, None},
    {{-3*Pi, -2*Pi, -Pi, 0, Pi, 2*Pi, 3*Pi}, None}},
  FrameLabel -> {{"\[Theta]'(t)", None},
    {"\[Theta](t)", "constant energy levels for nonlinear pendulum model"}}, ImageSize -> 40
  LabelStyle -> 14, RotateLabel -> False]
```

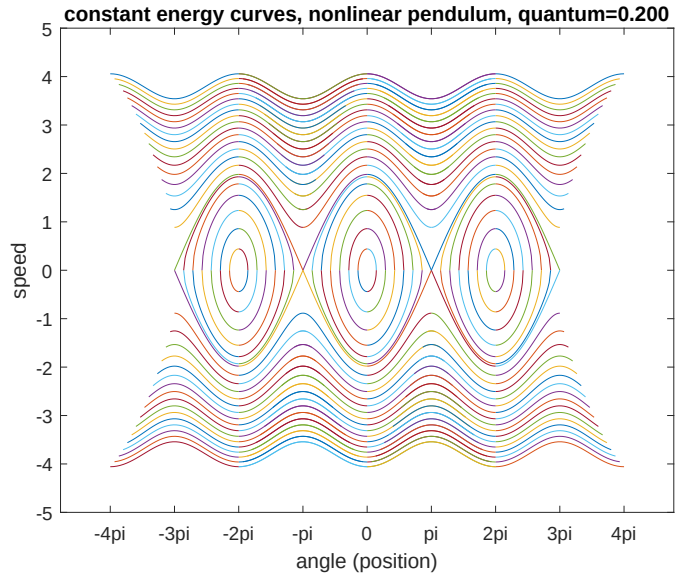


Matlab

```
function HW2
%HW2 problem. MAE 200A. UCI, Fall 2005
%by Nasser Abbasi

%This MATLAB function generate the constant energy level
%curves for a nonlinear pendulum with no damping

close all; clear;
```

```
m=1; g=9.8; L=10;

%number of data points (positio vs speed) to find per curve
nPoints=40;

nEnergyLevels = 8; %number of energy levels
lowAnglesToVisit = linspace(0,pi,nEnergyLevels);
lowEnergyLevels(1:nEnergyLevels)=m*g*L*(1-cos(lowAnglesToVisit));
highAnglesToVisit = linspace(pi,2*pi,2*nEnergyLevels);
highEnergyLevels=zeros(1,2*nEnergyLevels);

initialHighEnergy=2*m*g*L;
Q=0.2;
for i=1:2*nEnergyLevels
  highEnergyLevels(i) = initialHighEnergy+(Q*i*initialHighEnergy);
end

A = zeros(length(lowAnglesToVisit)+length(highAnglesToVisit),2);
A(:,1) = [lowAnglesToVisit highAnglesToVisit];
A(:,2) = [lowEnergyLevels highEnergyLevels];

[nAngles,~]=size(A);
data=zeros(nPoints,2);

for j=1:nAngles

    currentAngle=A(j,1);
    currentEnergyLevel =A(j,2) ;
    angles=linspace(0,currentAngle,nPoints);
    data(1:nPoints,1)=angles(:);

    for m=1:nPoints
        data(m,2)=speed(currentEnergyLevel,angles(m));
    end
    doPlots(data);
end

title(sprintf(['constant energy curves, nonlinear pendulum,'...
               'quantum=%1.3f'],Q));
xlabel('angle (position)');
ylabel('speed');
```

```
set(gca,'xtick',[-4*pi,-3*pi,-2*pi,-pi,0,pi,2*pi,3*pi,4*pi]);
set(gca,'XTickLabel',{'-4pi','-3pi','-2pi','-pi',...
                      '0','pi','2pi','3pi','4pi'})

print(gcf, '-dpdf', '-r600', 'images/matlab_e77_1');

function s=speed(energy,angle)
m=1; g=9.8; L=10;
if angle<pi
    s=sqrt(2/(m*L^2)*(energy-m*g*L*(1-cos(angle))));
else
    s=sqrt(2/(m*L^2)*(energy-m*g*L*(1+cos(angle-pi))));
end

function doPlots(data)
plotCurves(data,0);
plotCurves(data,2*pi);
plotCurves(data,-2*pi);

function plotCurves(data,k)
plot(data(:,1)+k,data(:,2));
hold on;
plot(data(:,1)+k,-data(:,2));
plot(-data(:,1)+k,-data(:,2));
plot(-data(:,1)+k,data(:,2));
```

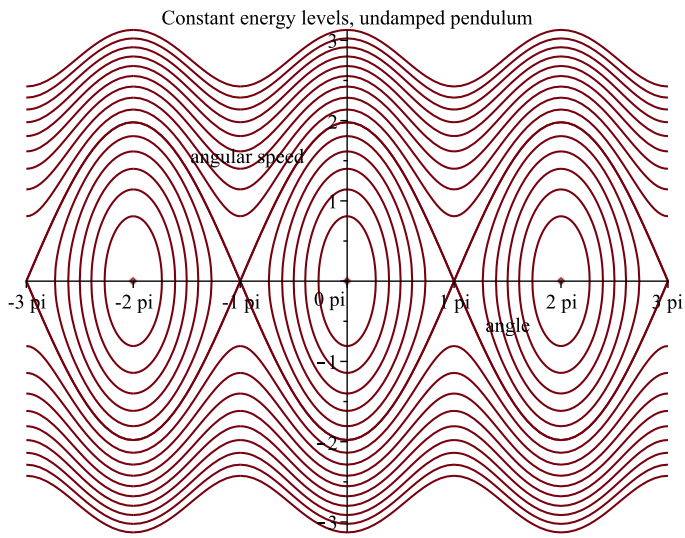


Maple

The Maple solution was contributed by Matrin Eisenberg

```
restart;
plotEnergyLevels:= proc(m::positive, g::positive, L::positive, Erange::positive, numContour
  local plotOptions, maxTheta, thetaDot, plotContour, Emax, energies, contours;
  plotOptions := args[6..-1];
  maxTheta := E- arccos(max(1-E/m/g/L, -1));
  thetaDot := E- theta- sqrt(max(2/m/L^2 * (E - m*g*L*(1-cos(theta))), 0));
  plotContour := E- plot(thetaDot(E), 0..maxTheta(E), numpoints=5, plotOptions);

  # Create first quadrant of middle region of the graph.
  Emax := Erange*m*g*L;
  energies := {Emax * n/numContours $ n=1..numContours};
  if Erange  2 then energies := energies union {2*m*g*L}; fi;
  contours := map(plotContour, energies);

  # Create other quadrants.
  map2(rcurry, plottools[reflect], {[[0,0],[0,1]], [0,0], [[0,0],[1,0]]});
  contours := contours union map(f- map(f, contours)[], %);

  # Create left and right regions of the graph.
  map2(rcurry, plottools[translate], {-2*Pi,2*Pi}, 0);
  contours := contours union map(f- map(f, contours)[], %);
```

```
  # Complete the graph.
  plots[display]([%[], plot([[-2*Pi,0], [0,0], [2*Pi,0]], plotOptions, style=point)],
    view=[-3*Pi..3*Pi, -thetaDot(Emax)(0)..thetaDot(Emax)(0)],
    title="Constant energy levels, undamped pendulum",
    labels=["angle", "angular speed"],
    xtickmarks=[seq](evalf(k*Pi)=sprintf("%a pi", k), k=-3..3));
end:
plotEnergyLevels(1, 9.8, 10, 5, 15);



;
```



sectionSolve numerically the ODE u""+u=f using point collocation method

Problem: Give the ODE

$$\frac{d^4u(x)}{dx^4} + u(x) = f$$

Solve numerically using the point collocation method. Use 5 points and 5 basis functions. Use the Boundary conditions $u(0) = u(1) = u''(0) = u''(1) = 0$

Use the trial function

$$g(x) = \sum_{i=1}^{N=5} a_i (x(x-1))^i$$

Use $f = 1$

The solution is approximated using $u(x) \approx g(x) = \sum_{i=1}^{N=5} a_i (x(x-1))^i$.

N equations are generated for N unknowns (the unknowns being the undetermined coefficients of the basis functions). This is done by setting the error to zero at those points. The error being

$$\frac{d^4 g(x)}{dx^4} + g(x) - f$$

Once $g(x)$ (the trial function is found) the analytical solution is used to compare with the numerical solution.

Mathematica

```
Clear["Global`*"];
nBasis  = 5;
nPoints = 5;
a = Array[v,{nBasis}]
```

```
Out[392]= {v[1],v[2],v[3],v[4],v[5]}
```

```
trial    = Sum[a[[n]] (x(x-1))^n,{n,1,nBasis}];
residual = D[trial,{x,4}]+trial-1;
mat      = Flatten[Table[residual==0/.x->n/(2*nPoints-1),
             {n,1,nPoints-1}]];

mat = Join[mat,{2 a[[1]]+2a[[2]]==0}];
sol = N@First@Solve[mat,a]
```

```
Out[397]= {v[1.]->-0.0412493,
           v[2.]->0.0412493,
           v[3.]->0.000147289,
           v[4.]->-0.0000245233,
           v[5.]->-3.28259*10^-8}
```

```
numericalSolution=Chop@
   FullSimplify@Sum[sol[[n,2]](x(x-1))^n,
                    {n,1,nBasis}]

numericalSolutionDiff4 = D[numericalSolution,{x,4}];
numericalMoment = D[numericalSolution,{x,2}];
```

$$0.0412493x - 0.0826459x^3 + 0.0416667x^4 - 0.00034374x^5 - 1.51283 * 10^-8x^6$$
$$+0.0000984213x^7 - 0.0000248515x^8 + 1.64129 * 10^-7x^9 - 3.28259 * 10^-8x^10$$

```
(*now analytical solution is obtained
using DSolve and compared to numerical solution*)
eq = u''''[x]+u[x]==1;
bc = {u[0]==0,u[1]==0,u''[0]==0,u''[1]==0};
analyticalSolution=First@DSolve[{eq,bc},u[x],x];
analyticalSolution=u[x]/.analyticalSolution;
analyticalSolution=FullSimplify[analyticalSolution]
```

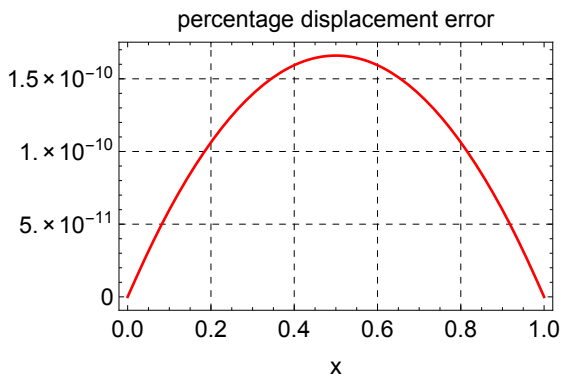
$$-\frac{\cos\left(\frac{1-(1+i)x}{\sqrt{2}}\right)+\cos\left(\frac{1-(1-i)x}{\sqrt{2}}\right)+\cosh\left(\frac{1-(1+i)x}{\sqrt{2}}\right)+\cosh\left(\frac{1-(1-i)x}{\sqrt{2}}\right)-2\left(\cos\left(\frac{1}{\sqrt{2}}\right)+\cosh\left(\frac{1}{\sqrt{2}}\right)\right)}{2\left(\cos\left(\frac{1}{\sqrt{2}}\right)+\cosh\left(\frac{1}{\sqrt{2}}\right)\right)}$$

```
analyticalMoment = D[analyticalSolution,{x,2}];
```

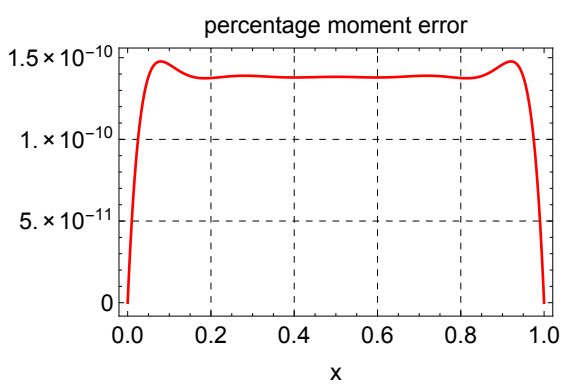
$$-\frac{-i\cos\left(\frac{1-(1+i)x}{\sqrt{2}}\right)+i\cos\left(\frac{1-(1-i)x}{\sqrt{2}}\right)+i\cosh\left(\frac{1-(1+i)x}{\sqrt{2}}\right)-i\cosh\left(\frac{1-(1-i)x}{\sqrt{2}}\right)}{2\left(\cos\left(\frac{1}{\sqrt{2}}\right)+\cosh\left(\frac{1}{\sqrt{2}}\right)\right)}$$

```
dispError[pt_] :=
  ((analyticalSolution-numericalSolution)/.x->pt)/
     (analyticalSolution/.x->.5)

momentError[pt_]:=
   ((analyticalMoment-numericalMoment)/.x->pt)/
       (analyticalMoment/.x->.5)

(*Now the percentage displacement
and moment errors are plotted*)
Plot[dispError[z],{z,0,1},
  ImagePadding->70,
  ImageSize->400,
  Frame->True,
  GridLines->Automatic,
```

```
  GridLinesStyle->Dashed,
  PlotStyle->Red,
  FrameLabel->{{"error",None},
    {"x","percentage displacement error"}},
    LabelStyle->14]
```

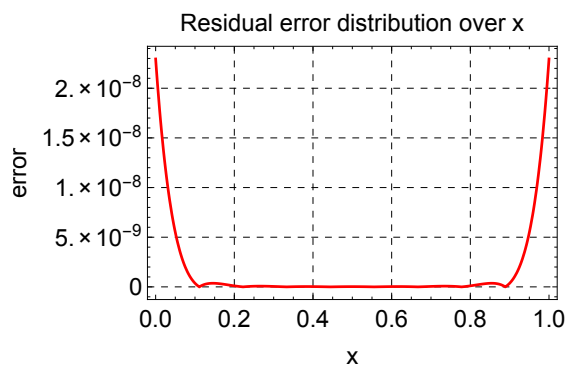



```
Plot[momentError[z],{z,0,1},
  ImagePadding->70,
  ImageSize->400,
  Frame->True,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  PlotStyle->Red,
  FrameLabel->{{"error",None},
    {"x","percentage moment error"}},
    LabelStyle->14,PlotRange->All]
```



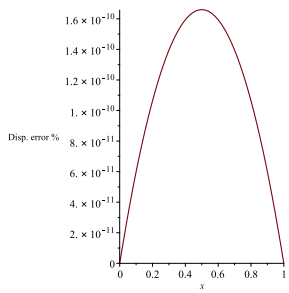

```
(*Now the Residual error distribution
```

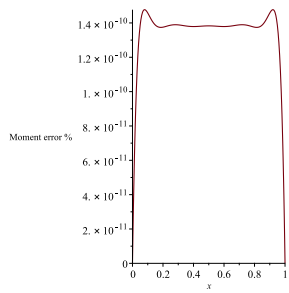
```
over x is plotted*)
Plot[Abs[numericalSolutionDiff4+
         numericalSolution-1],{x,0,1},
  ImagePadding->80,
  ImageSize->400,
  Frame->True,
  GridLines->Automatic,
  GridLinesStyle->Dashed,
  PlotStyle->Red,
  FrameLabel->{{"error",None},
   {"x","Residual error distribution over x"}},
   LabelStyle->14,
   PlotRange->All]
```



Maple

```
#Solve u''''+u=1 using point collocation. Zero B.C.
#written by Nasser Abbasi for fun on April 25,2006.
restart;
Digits:=15;
nBasis  := 5:
nPoints := 5:
coef    := [seq(a[i],i=1..nBasis)]:
g := (x,n)-coef[n]*(x*(x-1))^n:  #basis function
f := x-sum(g(x,k),k=1..nBasis):  #trial function
moment := x-diff(f(x),x$2):
residual := x-diff(f(x),x$4)+f(x)-1:
A := seq(subs(x=N/(2*nPoints-1),residual(x)=0),N=1..nPoints-1):
A := A,2*(coef[1]+coef[2]=0):
sol   := solve([A],coef):
coef  := map(rhs,sol[1]):
evalf(coef):
numericalSolution:=x-sum(coef[n]*(x*(x-1))^n  ,n=1..nBasis):
`The numerical solution is `, numericalSolution(x);
numericalSolutionMoment := x-diff(numericalSolution(x),x$2):

#Now obtain exact solution from Maple
eq    := diff(W(x),x$4)+W(x)=1:
bc    := W(0)=0,W(1)=0,D(D(W))(0)=0,D(D(W))(1)=0:

exact := unapply(rhs(dsolve({eq,bc})),x):

exactMoment := x-diff(exact(x),x$2):

displacmentError := x-(exact(x)-numericalSolution(x))/exact(.5):
momentError := x- (exactMoment(x)-numericalSolutionMoment(x))/evalf(subs(x=.5,exactMoment(x

plot(displacmentError(x),x=0..1,labels=['x',"Disp. error %"]);
plot(momentError(x),x=0..1,labels=['x',"Moment error %"]);
```





4.11 How to numerically solve a set of non-linear equations?

Numerically solve the following three equations for e, a, f

$$eq1 = r1 - a(e-1)$$
$$eq2 = delT - \sqrt{\frac{a^3}{\mu}}(e * \sinh(f) - f)$$
$$eq3 = r2 - a(e\cosh(f) - 1)$$

Mathematica

```
ClearAll[a, e, f]
delT = 5.215*60*60;
r1 = 200 + 6378;
r2 = 130000;
mu = 3.986*10^5;
eq1 = r1 - a*(e - 1) == 0;
eq2 = delT - Sqrt[a^3/mu]*
     (e*Sinh[f] - f) == 0;

eq3 = r2 - a*(e*Cosh[f]-1)==0;
sol = NSolve[{eq1, eq2, eq3},
           {a, e, f}, Reals]
```

```
{{a->12029.39633,
  e->1.546827108,
  f->2.721303232}}
```

Matlab

```
clear all
syms a e f;
delT = 5.215*60*60;
r1   = 200+6378;
r2   = 130000;
mu   = 3.986*10^5;
eq1  = r1 - a*(e-1);
eq2  = delT - sqrt(a^3/mu)*(e*sinh(f)-f);
eq3  = r2 - a*(e*cosh(f)-1);
sol = feval(symengine,'numeric::solve',...
          {eq1,eq2,eq3});
vpa(sol,6)
```

```
ans =
[a == 12029.4, e == 1.54683, f == 2.7213]
```

Another option is `solve` but slower

```
sol  = solve(eq1,eq2,eq3,a,e,f);
```

```
sol =
    a: [1x1 sym]
    e: [1x1 sym]
    f: [1x1 sym]

sol.a
12029.3963287144351264449270898 27

sol.e
1.5468271075497087492979481526009

sol.f
2.7213032317471583123822097902877
```

4.12 Solve 2nd order ODE (Van Der Pol) and generate phase plot

Problem: Solve

$$y''(t) - \mu\left(1 - y^2\right) y'(t) + y(t) = 0$$

for different values of μ to compare effect of changing μ on the solution trajectories. The initial conditions are

$$\begin{aligned} y(0) &= 0.1 \\ y'(t) &= 0 \end{aligned}$$

In both Mathematica and Matlab, numerical ODE solver was used.

For Matlab, The 2nd order ODE is first converted to 2 first order ODE's, and then solve the coupled ODE's using ode45. In Mathematica NDSolve was used. This does not require the conversion.

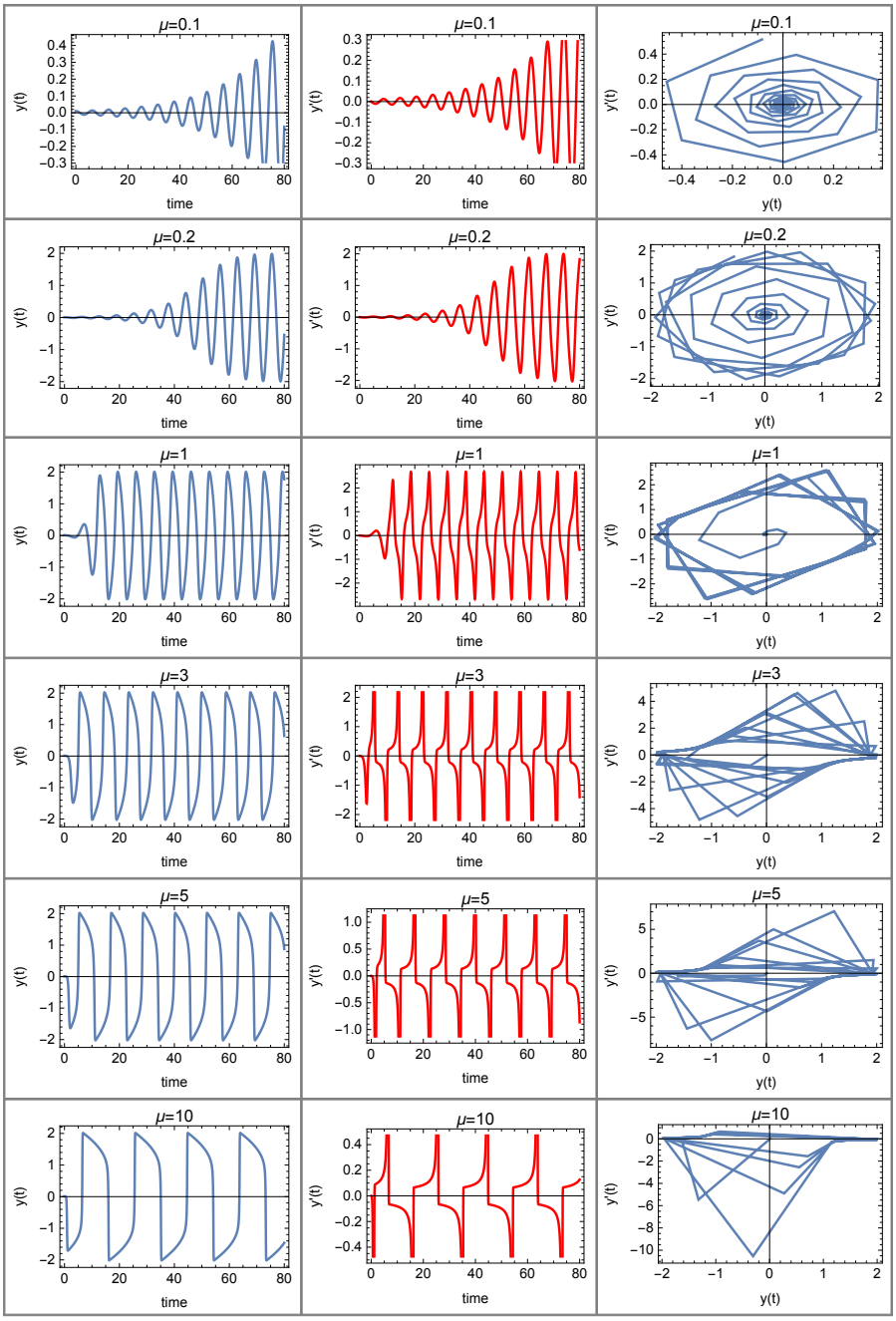
Starting by writing the 2nd order ODE as 2 first order ODE's

$$\left.\begin{aligned} x_1 &= y \\ x_2 &= y' \end{aligned}\right\} \text{derivatives} \Rightarrow \left.\begin{aligned} x_1' &= y' \\ x_2' &= y'' \end{aligned}\right\} \Rightarrow \left.\begin{aligned} x_1' &= x_2 \\ x_2' &= \mu(1 - x_1^2)\, x_2 + x_1 \end{aligned}\right\}$$

Below is the solution of the differential equation for different values of $\mu = 1$

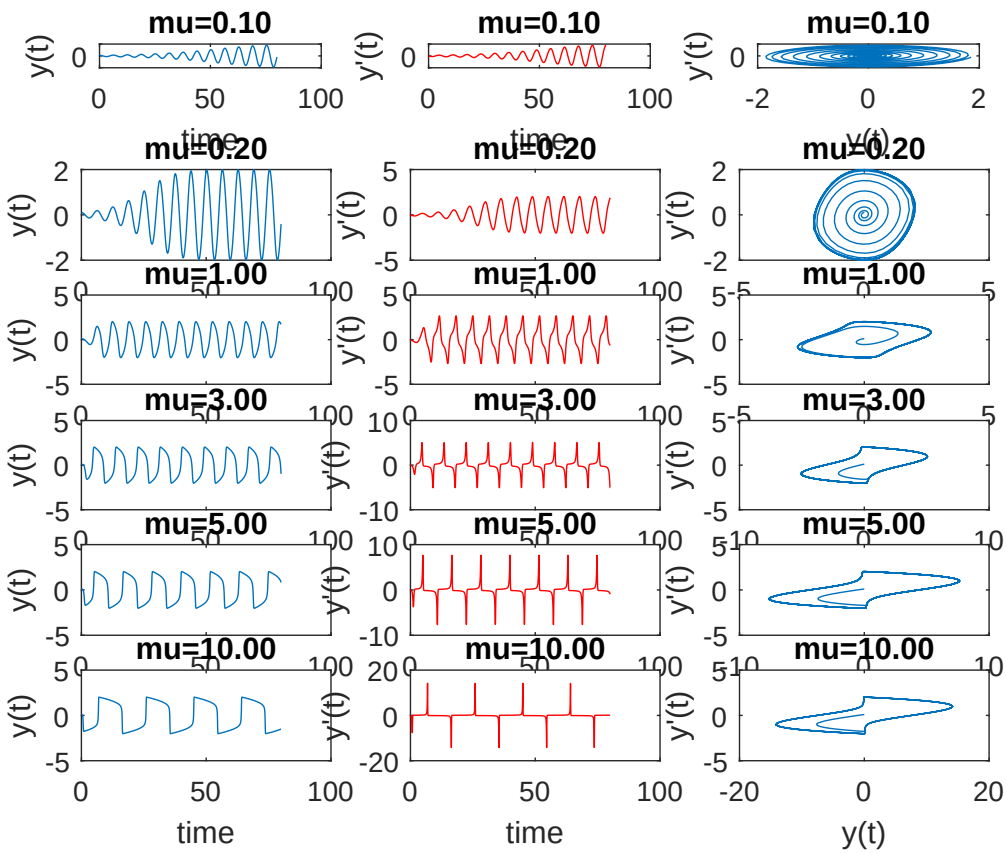
Mathematica

```
process[mu_]:=Module[
      {sol,p1,p2,p3,data,system,z,x1,
       x2,t,ode1,ode2,timeScale},
ode1 =x1'[t]==x2[t];
ode2 =x2'[t]==z(1-x1[t]^2)x2[t]-x1[t];
timeScale ={t,0,80};
sol=First@NDSolve[
  {ode1,ode2/.z->mu,x1[0]==0.01,x2[0]==0},
  {x1[t],x2[t]},
   timeScale,Method->{"Extrapolation",
   Method->"LinearlyImplicitEuler"}];
sol = {x1[t],x2[t]}/.sol;
p1 = Plot[Evaluate[sol[[1]]],
      Evaluate[timeScale],
      Frame->True,
      FrameLabel->{"time","y(t)"},
      PlotLabel->"\[Mu]="<>ToString[mu]];
p2 = Plot[Evaluate[sol[[2]]],
          Evaluate[timeScale],
      Frame->True,
      FrameLabel->{"time","y'(t)"},
      PlotLabel->"\[Mu]="<>ToString[mu],
      PlotStyle->RGBColor[1,0,0]];
data = Table[{evaluate[sol[[1]]],
        Evaluate[sol[[2]]]},
        Evaluate[timeScale]];
p3=ListPlot[data,Joined->True,Frame->True,
     PlotLabel->"\[Mu]="<>ToString[mu],
     FrameLabel->{"y(t)","y'(t)"},
     PlotRange->All];
{p1,p2,p3}
];
mu={0.1,.2,1,3,5,10};
r = process/@mu; Grid[r]
```


μ=0.1
μ=0.2
μ=1
μ=3
μ=5
μ=10
y(t)
y'(t)
time

Matlab

```
function e71
mu=[0.1,0.2,1,3,5,10];

for n=1:length(mu)
    process(mu(n),n-1);
end

function process(mu,n)
timeScale=[0 80];
ic=[0.1;0];

[t,x]=ode45(@ode,timeScale,ic,[],mu);
subplot(6,3,(3*n)+1);
plot(t,x(:,1));
title(sprintf('mu=%1.2f',mu));
xlabel('time'); ylabel('y(t)');

subplot(6,3,(3*n)+2);
plot(t,x(:,2),'r');
title(sprintf('mu=%1.2f',mu));
xlabel('time'); ylabel('y''(t)');

subplot(6,3,(3*n)+3);
plot(x(:,2),x(:,1));
title(sprintf('mu=%1.2f',mu));
xlabel('y(t)'); ylabel('y''(t)');


function xdot=ode(t,x,mu)
xdot=[x(2) ; mu*(1-x(1)^2)*x(2)-x(1)];
```



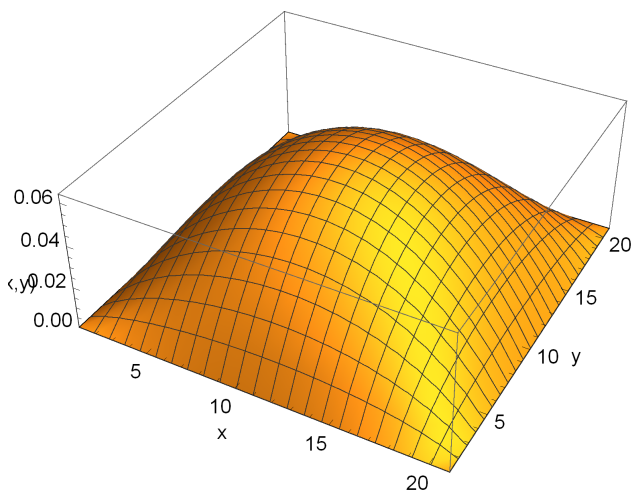
4.13 How to numerically solve Poisson PDE on 2D using Jacobi iteration method?

Problem: Solve $\nabla^2 u = f(x, y)$ on 2D using Jacobi method. Assume $f(x, y) = -\mathrm{e}^{-(x-0.25)^2-(y-0.6)^2}$. Use mesh grid norm and relative residual to stop the iterations.

Mathematica

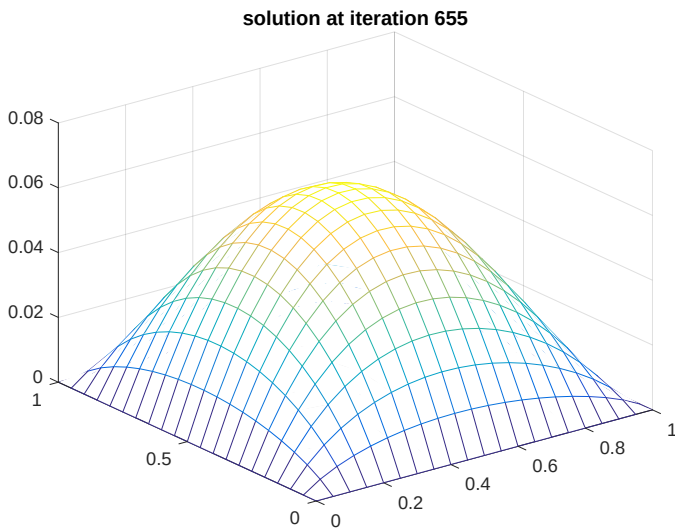
```
n = 21; (*number of grid points*)
uNew = Table[0, {n}, {n}]; (*solution grid*)
residual = uOld = uNew;

h = 1/(n - 1); (*spacing*)
c = 0.1;  (*controls the tolerance*)
epsilon = c*h^2; (*error tolerance*)

grid = N@Table[{i*h,j*h},{i,0,n-1},{j,0,n-1}];

(*the force function*)
f[x_,y_] := -Exp[-(x - 0.25)^2 - (y - 0.6)^2];

(*evaluate force on the grid*)
force = Map[f[#[[1]],#[[2]]]&,grid,{2}];
normF = Sqrt[h]*Norm[Flatten@force,2];(*force norm*)
converged = False;

k = 0; (*iteration counter*)
While[Not[converged],
 k++;
 For[i = 2, i < n, i++,
  For[j = 2, j < n, j++,
   uNew[[i, j]] = (1/4)*(uOld[[i-1,j]]+uOld[[i+1,j]]
    + uOld[[i,j-1]]+uOld[[i,j+1]]-h^2*force[[i,j]]);

   residual[[i,j]] = force[[i, j]]-1/h^2*(uOld[[i-1,j]]+
    uOld[[i+1,j]]+uOld[[i,j-1]]+uOld[[i,j+1]]-4*uOld[[i,j]])
   ]
 ];
 uOld = uNew; (*updated at end*)
 normR = Sqrt[h] Norm[Flatten@residual, 2];
 If[normR/normF < epsilon, converged = True]
];

(*plot solution*)
ListPlot3D[uOld, PlotRange -> All, ImagePadding -> 30,
  ImageSize -> 400, Mesh -> {n, n},
  AxesLabel -> {"x", "y", "u(x,y)"},
  PlotRangePadding -> 0, BaseStyle -> 12,
  PlotLabel -> Row[{"Converged in ", k, " iterations",
            " ,epsilon=", epsilon, " ,h=", h}]]
```

Converged in 655 iterations ,epsilon=0.00025 ,h=$\frac{1}{20}$



Matlab

```
close all; clear all;
n = 21; % grid points, including internal points
c = 0.1; % constant of tolerance
myforce = @(X,Y) -exp( -(X-0.25).^2 - (Y-0.6).^2 );

h     = 1/(n-1);        % grid spacing
tol   = c * h^2;        % tolerance
res   = zeros(n,n);     % storage for residual
u     = res;            % storage for solution
uNew  = u;
k     = 0;                      % loop counter
[X,Y] = meshgrid(0:h:1,0:h:1); % coordinates
f     = myforce(X,Y);
normf = norm( f(:),2); % find norm

%Now start the iteration solver, stop when
%relative residual < tolerance
figure;
i     = 2:n-1;
j     = 2:n-1;      %the iterations vectorized
done  = false;

while ~done
    k = k+1;

    uNew(i,j) = (1/4)*( u(i-1,j) + u(i+1,j) + ...
                  u(i,j-1) + u(i,j+1) - h^2 * f(i,j) );
    res(i,j) = f(i,j) - (1/h^2)*( u(i-1,j) + ...
           u(i+1,j) + u(i,j-1) + u(i,j+1) - 4*u(i,j) );

    %uncomment to see it update as it runs

    %mesh(X,Y,u);  %hold on;
    %title(sprintf('solution at iteration %d',k));
    %zlim([0,.07]);
    %drawnow;

    if norm(res(:),2)/normf < tol
        done = true;
    end;

    u = uNew;
end

mesh(X,Y,u);
title(sprintf('solution at iteration %d',k))
```



4.14 How to solve BVP second order ODE using finite elements with linear shape functions and using weak form formulation?

Solve $y''(x) - 3y(x) = -x^2$ over $x = 0 \ldots 1$ with boundary conditions $x(0) = 0$ and $x(1) = 0$ using piecewise linear trial functions.

solution

Let the trial function be the linear function $\hat{y}(x) = c_1 x + c_2$. The residual is $R = \hat{y}''(x) - 3\hat{y}(x) + x^2$. Let the test function be $w(x)$. We need to solve for each element i the following equation

$$I_i = \int w_i \, R_i \, dx$$

Using the weak form, we apply integration by parts to obtain

$$\begin{aligned} I_i &= \int_{x_i}^{x_{i+1}} -\frac{dw}{dx}\frac{d\hat{y}}{dx} - 3w(x)\hat{y}(x) + w(x)x^2 \, dx + \left[w(x)\frac{d\hat{y}}{dx} \right]_{x_i}^{x_{i+1}} \\ &= 0 \end{aligned} \tag{96.1}$$

or

$$
\begin{aligned}
I &= \sum_{i=1}^{i=M} I_i \\
&= \sum_{i=1}^{i=M} \int_{x_i}^{x_{i+1}} \left(-\frac{dw}{dx}\frac{d\hat{y}}{dx} - 3w(x)\hat{y}(x) + w(x)x^2\,dx + \left[w(x)\frac{d\hat{y}}{dx} \right]_{x_i}^{x_{i+1}} \right) \\
&= 0
\end{aligned}
$$

Where M is the number of elements. The above is the M equations we need to solve for the unknowns y_i at the internal nodes. In Galerkin method, the test function is $w_1(x) = H_1(x)$ and $w_2(x) = H_2(x)$ which comes from writing $w_i(x) = \frac{d\hat{y}(x)}{dy_i}$

Rewriting the trial function $\hat{y}(x)$ in terms of unknown values at nodes results in

$$\hat{y}(x) = H_1(x)y_i + H_2(x)y_{i+1}$$

Where $H_1(x) = \frac{x_{i+1}-x}{h}$ and $H_2(x) = \frac{x-x_i}{h}$ where h is the length of each element. Hence (96.1) becomes

$$I_i = \int_{x_i}^{x_{i+1}} -\begin{Bmatrix} H_1'(x) \\ H_2'(x) \end{Bmatrix} \left\{ H_1'(x)H_2'(x) \right\} \begin{Bmatrix} y_i \\ y_{i+1} \end{Bmatrix} - 3\begin{Bmatrix} H_1(x) \\ H_2(x) \end{Bmatrix} \left\{ H_1(x)H_2(x) \right\} \begin{Bmatrix} y_i \\ y_{i+1} \end{Bmatrix} + \begin{Bmatrix} H_1(x) \\ H_2(x) \end{Bmatrix} x^2\,dx + \tag{96.2}$$

The terms $\left[w(x)\frac{d\hat{y}}{dx} \right]_{x_i}^{x_{i+1}}$ reduce to $\left[w(x)\frac{d\hat{y}}{dx} \right]_0^1$ since intermediate values cancel each other leaving the two edge values over the whole domain.

But $H_1'(x) = \frac{-x}{h}$ and $H_2'(x) = \frac{x}{h}$. Plugging these into (96.2) and integrating gives the following

$$I_i = \frac{1}{h}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{Bmatrix} y_i \\ y_{i+1} \end{Bmatrix} - h\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix} \begin{Bmatrix} y_i \\ y_{i+1} \end{Bmatrix} + \frac{1}{12h}\begin{bmatrix} 3x_i^4 - 4x_i^3x_{i+1} + x_{i+1}^4 \\ x_i^4 - 4x_ix_{i+1}^3 + 3x_{i+1}^4 \end{bmatrix} + \left[w(x)\frac{d\hat{y}}{dx} \right]_0^1 \tag{96.3}$$

The algebra above was done with Mathematica. In the following code, x_1 is x_i and x_2 is x_{i+1}

```
h1 = (x2 - x)/h;
h2 = (x - x1)/h;
int1 = -{{D[h1, x]}, {D[h2, x]}}.{{D[h1, x], D[h2, x]}};
int2 = -3 {{h1}, {h2}}.{{h1, h2}};
int3 = {{h1}, {h2}} x^2;
sol1 = Integrate[int1, {x, x1, x2}];
sol1 /. {(x1 - x2) -> -h, (x2 - x1) -> h}
```

```
{{-(1/h), 1/h}, {1/h, -(1/h)}}
```

```
sol2 = Integrate[int2, {x, x1, x2}];
sol2 /. {(x1 - x2) -> -h, (x2 - x1) -> h}
```

```
{{-h, -(h/2)}, {-(h/2), -h}}
```

```
sol3 = Integrate[int3, {x, x1, x2}];
Simplify[%];
% /. {(x1 - x2) -> -h, (x2 - x1) -> h}
```

```
{{(3 x1^4 - 4 x1^3 x2 + x2^4)/(12 h)},
 {(x1^4 - 4 x1 x2^3 + 3 x2^4)/(12 h)}}
```

The above equation I_i is calculated for each element i, resulting in M equations where M is the number of elements. Collecting all these local equations into a global stiffness matrix and then adjusting the global stiffness matrix for boundary conditions by eliminating corresponding rows and columns, it is then solved for the unknowns y_i as a system $Ax = f$ using direct solver. The following code illustrates the method.

Matlab

```
M   = 9;  %number of elements
N   = M+1; %number of nodes
dof = 2; %degree of freedom per element
kk  = zeros(N);
f   = zeros(N,1); %force vector
h   = 1/M; %length of each element
k_local=1/h*[-1 1;1 -1]-h*[1 1/2;1/2 1];

for i = 1:M  %loop over all elements
 x1 = (i-1)*h;
 x2 = i*h;
 f_local = -1/(12*h)*[3*x1^4-4*x1^3*x2+...
       x2^4;x1^4-4*x1*x2^3+3*x2^4];
 f(i:i+dof-1) = f(i:i+dof-1)+f_local;
 kk(i:i+dof-1,i:i+dof-1) = ...
    kk(i:i+dof-1,i:i+dof-1)+k_local;
end
%fix first and last rows since these are B.C.
kk(1,:)     = 0;
kk(1,1)     = 1;
kk(end,:)   = 0;
kk(end,end) = 1;
f(1)        = 0;
f(end)      = 0;
y           = kk\f; %solve

plot(0:h:1,y,'or')
hold on;
plot(0:h:1,y,'-');
title(sprintf...
 ('FEM solution using to ODE using %d elements',M'));
xlabel('x');
ylabel('y(x)');
grid
```

The analytical solution to the ODE is below. We can see it is very close to the FEM solution above with 11 elements. More elements leads to more accurate solution.

Mathematica

Analtical solution

```
ClearAll[y, x]
sol= y[x]/.First@DSolve[{y''[x]-3 y[x]==-x^2,
        y[0] == 0, y[1] == 0}, y[x], x];
Plot[sol, {x, 0, 1}, Frame -> True,
      FrameLabel -> {{"y(x)", None},
          {"x", "analytical solution"}},
      GridLines -> Automatic,
      GridLinesStyle -> {{Dashed, LightGray},
                         {Dashed, LightGray}}
    ]
```

Numerical

```
m = 9;
n = m + 1;
dof = 2;
kk = Table[0., {n}, {n}];
f = Table[0., {n}];
h = 1/m;
kLocal = 1/h {{-1, 1}, {1, -1}} - h {{1, 1/2}, {1/2, 1}};
Do[
  x1 = (i - 1) h;
  x2 = i h;
  fLocal = -1/(12 h) {3 x1^4 - 4 x1^3 x2 + x2^4,
              x1^4 - 4 x1 x2^3 + 3 x2^4};
  f[[i ;; i + dof - 1]] += fLocal;
  kk[[i ;; i + dof - 1, i ;; i + dof - 1]] +=kLocal,
  {i, m}
  ];
kk[[1, ;;]] = 0;
kk[[1, 1]] = 1;
kk[[-1, ;;]] = 0;
kk[[-1, -1]] = 1;
f[[1]] = 0;
f[[-1]] = 0;
y = LinearSolve[kk, f];
x = Range[0, 1, h];

ListPlot[Transpose[{x, y}], Frame -> True, Joined -> True,
         PlotMarkers -> Automatic, FrameLabel ->
```

```
            {{"y(x)", None}, {"x",
 Row[{"solution to y''(x)-3 y(x)=-x^2 using FEM, 9 elements"}]}},
 GridLines -> Automatic,
 GridLinesStyle -> LightGray
]
```

4.15 How to solve Poisson PDE in 2D using finite elements methods using rectangular element?

Solve $\nabla^2 u = f(x, y)$ on square using FEM. Assume $u = 0$ on boundaries and solve using $f(x, y) = xy$ and also $f(x, y) = 1$ and $f(x, y) = 3(\cos(4\pi x) + \sin(3\pi y))$

Use Galerkin method and weak form, Using a bilinear trial function. Let width of square be 1.

Solution

Using as an example with 9 elements to illustrate the method. The program below can be called with different number of elements.

The trial function is bilinear

$$\tilde{u} = c_1 + c_2 x + c_3 y + c_4 xy \tag{1}$$

Looking at one element, and using local coordinates systems with element having width $2a$ and height $2b$ gives

Evaluating the trial function (1) at each corner node of the above element gives

$$\begin{aligned}
\tilde{u}_1 &= c_1 - ac_2 - bc_3 + abc_4 \\
\tilde{u}_2 &= c_1 + ac_2 - bc_3 - abc_4 \\
\tilde{u}_3 &= c_1 + ac_2 + by_3 + abc_4 \\
\tilde{u}_4 &= c_1 - ac_2 + bc_3 + abc_4
\end{aligned}$$

Or

$$\begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} = \begin{pmatrix} 1 & -a & -b & ab \\ 1 & a & -b & -ab \\ 1 & a & b & ab \\ 1 & -a & b & ab \end{pmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix}$$

Hence

$$\begin{aligned} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} &= \begin{pmatrix} 1 & -a & -b & ab \\ 1 & a & -b & -ab \\ 1 & a & b & ab \\ 1 & -a & b & ab \end{pmatrix}^{-1} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} \\ &= \frac{1}{4ab} \begin{pmatrix} ab & ab & ab & ab \\ -b & b & b & -b \\ -a & -a & a & a \\ 1 & -1 & 1 & -1 \end{pmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} \end{aligned} \tag{2}$$

```
coor = {{-a, -b}, {a, -b}, {a, b}, {-a, b}};
uVal = {u1, u2, u3, u4}; eq = c1 + c2 x + c3 y + c4 x y;
r = MapThread[eq /. {u -> #1, x -> #2[[1]], y -> #2[[2]]}&, {uVal, coor}];
mat = Last@Normal[CoefficientArrays[r, {c1, c2, c3, c4}]];
MatrixForm[Inverse[mat] // MatrixForm
```

Substituting (2) back into (1) and rearranging terms results in

$$\begin{aligned} \tilde{u} &= c_1 + c_2 x + c_3 y + c_4 xy \\ &= \frac{1}{4ab}\left((ab - bx - ay + xy)\,\tilde{u}_1 + (ab + bx - ay - xy)\,\tilde{u}_2 + (ab + bx + ay + xy) + (ab - bx + ay - xy)\right) \end{aligned}$$

Since ab is the $\frac{1}{4}$ of the area of element, then the above becomes

$$\tilde{u} = \frac{1}{A}((ab - bx - ay + xy)\,\tilde{u}_1 + (ab + bx - ay - xy)\,\tilde{u}_2 + (ab + bx + ay + xy) + (ab - bx + ay - xy))$$

The above can now be written in term of what is called shape functions

$$\tilde{u}(x,y) = N_1(x,y)\tilde{u_1} + N_2(x,y)\tilde{u_2}N_3(x,y)\tilde{u_3} + N_4(x,y)\tilde{u_4}$$

Where

$$\begin{aligned}
N_1 &= \frac{1}{A}(ab - bx - ay + xy) = \frac{1}{A}(a - x)\,(b - y)\\
N_2 &= \frac{1}{A}(ab + bx - ay - xy) = \frac{1}{A}(a + x)\,(b - y)\\
N_3 &= \frac{1}{A}(ab + bx + ay + xy) = \frac{1}{A}(a + x)\,(b + y)\\
N_4 &= \frac{1}{A}(ab - bx + ay - xy) = \frac{1}{A}(a - x)\,(b + y)
\end{aligned}$$

Now that the shape functions are found, the next step is to determine the local element stiffness matrix. This can be found from the weak form integral over the area of the element.

$$I_i = \int_{x=-a}^{x=a}\int_{y=-b}^{y=b} w\big(\nabla^2 u - f(x,y)\big)\;dxdy \tag{3}$$

Where $w(x,y)$ is the test function. For Galerkin method, the test function $w_i = \frac{d\tilde{u}}{du_i} = N_i(x,y)$. Hence

$$w(x,y) = \begin{Bmatrix} N_1(x,y)\\ N_2(x,y)\\ N_3(x,y)\\ N_4(x,y)\end{Bmatrix} = \begin{Bmatrix} \frac{1}{A}(a - x)\,(b - y)\\ \frac{1}{A}(a + x)\,(b - y)\\ \frac{1}{A}(a + x)\,(b + y)\\ \frac{1}{A}(a - x)\,(b + y)\end{Bmatrix}$$

Using weak form, integration by parts is applied to (2)

$$I_i = \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \left(-\frac{\partial w}{\partial x_i} \frac{\partial \tilde{u}}{\partial x_i} - w f(x,y) \right) dxdy + \overbrace{\int_{\Gamma} w \frac{\partial \tilde{u}}{\partial n}}^{\text{goes to zero}}$$

The term $\int_{\Gamma} w \frac{\partial \tilde{u}}{\partial n}$ is the integration over the boundary of the element. Since there is only an essential boundary condition over all the boundaries (this is the given Dirchilet boundary condition), $w = 0$ on the boundary and this integral vanishes. There is no natural boundary conditions for this example.

For those elements not on the external edge of the overall grid (i.e. internal elements), each contribution to this integral will cancel from the adjacent internal element. What this means is that the above reduces to just the first integral

$$\begin{aligned}
I_i &= \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \left(-\frac{\partial w}{\partial x_i} \frac{\partial \tilde{u}}{\partial x_i} - w f(x,y) \right) dxdy \\
&= \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} -\begin{Bmatrix} \frac{\partial N_1}{\partial x} \\ \frac{\partial N_2}{\partial x} \\ \frac{\partial N_3}{\partial x} \\ \frac{\partial N_4}{\partial x} \end{Bmatrix} \begin{Bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial x} \end{Bmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} - \begin{Bmatrix} \frac{\partial N_1}{\partial y} \\ \frac{\partial N_2}{\partial y} \\ \frac{\partial N_3}{\partial y} \\ \frac{\partial N_4}{\partial y} \end{Bmatrix} \begin{Bmatrix} \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial y} \end{Bmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} \\
&- \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \begin{Bmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \\ N_4(x,y) \end{Bmatrix} f(x,y)\ dxdy
\end{aligned}$$

Hence

$$
\begin{aligned}
I_i = & \int_{x=-a}^{x=a}\int_{y=-b}^{y=b} -\begin{pmatrix}
\frac{\partial N_1}{\partial x}\frac{\partial N_1}{\partial x}+\frac{\partial N_1}{\partial y}\frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x}\frac{\partial N_2}{\partial x}+\frac{\partial N_1}{\partial y}\frac{\partial N_2}{\partial y} & \frac{\partial N_1}{\partial x}\frac{\partial N_3}{\partial x}+\frac{\partial N_1}{\partial y}\frac{\partial N_3}{\partial y} & \frac{\partial N_1}{\partial x}\frac{\partial N_4}{\partial x}+\frac{\partial N_1}{\partial y}\frac{\partial N_4}{\partial y}\\
\frac{\partial N_2}{\partial x}\frac{\partial N_1}{\partial x}+\frac{\partial N_2}{\partial y}\frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial x}\frac{\partial N_2}{\partial x}+\frac{\partial N_2}{\partial y}\frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x}\frac{\partial N_3}{\partial x}+\frac{\partial N_2}{\partial y}\frac{\partial N_3}{\partial y} & \frac{\partial N_2}{\partial x}\frac{\partial N_4}{\partial x}+\frac{\partial N_2}{\partial y}\frac{\partial N_4}{\partial y}\\
\frac{\partial N_3}{\partial x}\frac{\partial N_1}{\partial x}+\frac{\partial N_3}{\partial y}\frac{\partial N_1}{\partial y} & \frac{\partial N_3}{\partial x}\frac{\partial N_2}{\partial x}+\frac{\partial N_3}{\partial y}\frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial x}\frac{\partial N_3}{\partial x}+\frac{\partial N_3}{\partial y}\frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x}\frac{\partial N_4}{\partial x}+\frac{\partial N_3}{\partial y}\frac{\partial N_4}{\partial y}\\
\frac{\partial N_4}{\partial x}\frac{\partial N_1}{\partial x}+\frac{\partial N_4}{\partial y}\frac{\partial N_1}{\partial y} & \frac{\partial N_4}{\partial x}\frac{\partial N_2}{\partial x}+\frac{\partial N_4}{\partial y}\frac{\partial N_2}{\partial y} & \frac{\partial N_4}{\partial x}\frac{\partial N_3}{\partial x}+\frac{\partial N_4}{\partial y}\frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial x}\frac{\partial N_4}{\partial x}+\frac{\partial N_4}{\partial y}\frac{\partial N_4}{\partial y}
\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\\ \tilde{u}_4\end{Bmatrix}\\
& -\int_{x=-a}^{x=a}\int_{y=-b}^{y=b}\begin{Bmatrix}N_1(x,y)\\ N_2(x,y)\\ N_3(x,y)\\ N_4(x,y)\end{Bmatrix} f(x,y)\ dxdy
\end{aligned}
$$

In the above, we have the from $I_i = \int_\Omega -k_i\{\tilde{u}\}\, d\Omega - \int_\Omega \{\tilde{u}\}\, f_i d\Omega$, hence the element stiffness matrix is the first integral, and the force vector comes from the second integral.

The integration is now carried out to obtain the element stiffness matrix. This was done using Mathematica. The local stiffness matrix for element i is

$$
\begin{aligned}
k_i &= \int_{x=-a}^{x=a}\int_{y=-b}^{y=b} \frac{\partial w}{\partial x_i}\frac{\partial \tilde{u}}{\partial x_i} dxdy\\
&= \begin{pmatrix}
\frac{a^2+b^2}{3ab} & \frac{a}{6b}-\frac{b}{3a} & -\frac{a^2+b^2}{6ab} & -\frac{a}{3b}+\frac{b}{6a}\\
\frac{a}{6b}-\frac{b}{3a} & \frac{a^2+b^2}{3ab} & -\frac{a}{3b}+\frac{b}{6a} & -\frac{a^2+b^2}{6ab}\\
-\frac{a^2+b^2}{6ab} & -\frac{a}{3b}+\frac{b}{6a} & \frac{a^2+b^2}{3ab} & \frac{a}{6b}-\frac{b}{3a}\\
-\frac{a}{3b}+\frac{b}{6a} & -\frac{a^2+b^2}{6ab} & \frac{a}{6b}-\frac{b}{3a} & \frac{a^2+b^2}{3ab}
\end{pmatrix}
\end{aligned}
$$

For example, for element of width 1 and height 1, then $a = \frac{1}{2}, b = \frac{1}{2}$ and the above becomes

$$
k_i = \begin{pmatrix}
\frac{2}{3} & -\frac{1}{6} & -\frac{1}{3} & -\frac{1}{6}\\
-\frac{1}{6} & \frac{2}{3} & -\frac{1}{6} & -\frac{1}{3}\\
-\frac{1}{3} & -\frac{1}{6} & \frac{2}{3} & -\frac{1}{6}\\
-\frac{1}{6} & -\frac{1}{3} & -\frac{1}{6} & \frac{2}{3}
\end{pmatrix}
$$

```
ClearAll[a, b];
area = 4 a b;
phi = (1/area) {(a - x) (b - y),
                (a + x) (b - y),
                (a + x) (b + y),
                (a - x) (b + y)};
vx = {{D[phi[[1]], x]},
      {D[phi[[2]], x]},
      {D[phi[[3]], x]},
      {D[phi[[4]], x]}};
vy = {{D[phi[[1]], y]},
      {D[phi[[2]], y]},
      {D[phi[[3]], y]},
      {D[phi[[4]], y]}};
k = Integrate[vx.Transpose[vx]+vy.Transpose[vy],{x, -a, a}, {y, -b, b}]
```

Hence

$$I_i = -k_i \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \end{Bmatrix} - \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \begin{Bmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \\ N_4(x,y) \end{Bmatrix} f(x,y)\ dxdy$$

Now the integration of the force vector is carried out.

$$
\begin{aligned}
I_i^f &= \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \begin{Bmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \\ N_4(x,y) \end{Bmatrix} f(x,y)\ dxdy \\
&= \int_{x=-a}^{x=a} \int_{y=-b}^{y=b} \begin{Bmatrix} \frac{1}{A}(a-x)\,(b-y) \\ \frac{1}{A}(a+x)\,(b-y) \\ \frac{1}{A}(a+x)\,(b+y) \\ \frac{1}{A}(a-x)\,(b+y) \end{Bmatrix} f(x,y)\ dxdy
\end{aligned}
$$

In the above, the integration depends on the physical location of the element. We used a local coordinate system initially to obtain the shape functions. This has now to be transformed to the global coordinates system. Taking the center of each element as (x_0, y_0) then we need to replace $a \to x_0 + a$ and $b \to y_0 + b$ everywhere in the above integral. We did not have to do this for finding the local stiffness matrix, since that did not depend on the physical location of the element (it was constant). But for the integration of the force function, we need to do this mapping. In the code, the center of each element is found, and the replacement is done.

4.15.1 Integrating the force vector

This is normally done using Gaussian quadrature method. In this example, the integral is found off-line using Mathematica.

Evaluating the force vector requires integration over the element. This was done off-line using Mathematica. The result is a function of the coordinate of the center of the element. Then when running the code, this was evaluated for each element in the main loop.

4.15.1.1 $f(x,y) = xy$

For $f(x,y) = xy$ the above gives a result as function of the center of the physical element.

$$
I_i^f = \int_{x=(x_0-a)}^{x=x_0+a} \int_{y=(y_0-b)}^{y=y_0+b} \begin{Bmatrix} \frac{1}{A}((x_0-a)-x)\,(y_0+b-y) \\ \frac{1}{A}((x_0-a)+x)\,(y_0+b-y) \\ \frac{1}{A}((x_0-a)+x)\,(y_0+b+y) \\ \frac{1}{A}((x_0-a)-x)\,(y_0+b+y) \end{Bmatrix} xy\,dxdy
$$

This was done using Mathematica

$$I_i^f = \frac{1}{9\,(x_0+a)\,(y_0+b)} \begin{Bmatrix} a^2b^2(a-3x_0)\,(b-3y_0) \\ -ab^2(a^2+3ax_0+6x_0^2)\,(b-3y_0) \\ ab(a^2+3ax_0+6x_0^2)\,(b^2+3by_0+6y_0^2) \\ -ab^2(a-3x_0)\,(b^2+3by_0+y_0^2) \end{Bmatrix}$$

```
In[134]:= phi
Out[134]= {((a - x)*(b - y))/(4*a*b),
           ((a + x)*(b - y))/(4*a*b),
           ((a + x)*(b + y))/(4*a*b),
           ((a - x)*(b + y))/(4*a*b)}

In[135]:= phi /. {a :> x0 + a, b :> y0 + b}

Out[135]= {
  ((a - x + x0)*(b - y + y0))/(4*(a + x0)*(b + y0)),
  ((a + x + x0)*(b - y + y0))/(4*(a + x0)*(b + y0)),
  ((a + x + x0)*(b + y + y0))/(4*(a + x0)*(b + y0)),
  ((a - x + x0)*(b + y + y0))/(4*(a + x0)*(b + y0))}

Integrate[% x*y, {x, x0 - a, x0 + a}, {y, y0 - b, y0 + b}]

Out[136]= {
  (a^2*b^2*(a - 3*x0)*(b - 3*y0))/(9*(a + x0)*(b + y0)),
  -((a*b^2*(a^2 + 3*a*x0 + 6*x0^2)*(b - 3*y0))/(9*(a + x0)*(b + y0))),
  (a*b*(a^2 + 3*a*x0 + 6*x0^2)*(b^2 + 3*b*y0 + 6*y0^2))/(9*(a + x0)*(b + y0)),
  -((a^2*b*(a - 3*x0)*(b^2 + 3*b*y0 + 6*y0^2))/(9*(a + x0)*(b + y0)))}
```

4.15.2 Case 1

Case $f(x,y) = 3(\cos(4\pi x) + \sin(3\pi y))$

This was done using Mathematica

```
phi
phi /. {a :> (x0 + a), b :> (y0 + b)}
Integrate[%  3 (Cos[4 Pi x] + Sin[3 Pi y]),
   {x, x0 - a, x0 + a}, {y, y0 - b, y0 + b}]
```

```
{(1/(96*Pi^2*(a + x0)*(b + y0)))*(9*b^2*Cos[4*Pi*(-a + x0)] -
        9*b^2*Cos[4*Pi*(a + x0)] + 8*a*(12*a*b*Pi*Cos[3*Pi*(b - y0)] -
             9*b^2*Pi*Sin[4*Pi*(-a + x0)] - 2*a*(Sin[3*Pi*(b - y0)] +
                  Sin[3*Pi*(b + y0)]))), (1/(96*Pi^2*(a + x0)*(b + y0)))*
     (-9*b^2*Cos[4*Pi*(-a + x0)] + 9*b^2*Cos[4*Pi*(a + x0)] +
        8*(12*a*b*Pi*(a + 2*x0)*Cos[3*Pi*(b - y0)] -
             9*b^2*Pi*x0*Sin[4*Pi*(-a + x0)] + 9*a*b^2*Pi*Sin[4*Pi*(a + x0)] +
             9*b^2*Pi*x0*Sin[4*Pi*(a + x0)] - 2*a^2*Sin[3*Pi*(b - y0)] -
             4*a*x0*Sin[3*Pi*(b - y0)] - 2*a^2*Sin[3*Pi*(b + y0)] -
             4*a*x0*Sin[3*Pi*(b + y0)])), (1/(96*Pi^2*(a + x0)*(b + y0)))*
     (-9*b*(b + 2*y0)*Cos[4*Pi*(-a + x0)] + 9*b*(b + 2*y0)*Cos[4*Pi*(a + x0)] -
        72*b*Pi*x0*(b + 2*y0)*Sin[4*Pi*(-a + x0)] + 72*b*Pi*(a + x0)*(b + 2*y0)*
          Sin[4*Pi*(a + x0)] + 12*(a + x0)^2*(6*Pi*y0*Cos[3*Pi*(b - y0)] -
             6*Pi*(b + y0)*Cos[3*Pi*(b + y0)] + Sin[3*Pi*(b - y0)] +
             Sin[3*Pi*(b + y0)]) - 4*(-a + x0)*(a + 3*x0)*(6*Pi*y0*Cos[3*Pi*(b - y0)] -
             6*Pi*(b + y0)*Cos[3*Pi*(b + y0)] + Sin[3*Pi*(b - y0)] +
             Sin[3*Pi*(b + y0)])), (1/(96*Pi^2*(a + x0)*(b + y0)))*
     (9*b*(b + 2*y0)*Cos[4*Pi*(-a + x0)] - 9*b*(b + 2*y0)*Cos[4*Pi*(a + x0)] +
        8*a*(12*a*Pi*y0*Cos[3*Pi*(b - y0)] - 12*a*Pi*(b + y0)*Cos[3*Pi*(b + y0)] -
             9*b^2*Pi*Sin[4*Pi*(-a + x0)] - 18*b*Pi*y0*Sin[4*Pi*(-a + x0)] +
             2*a*Sin[3*Pi*(b - y0)] + 2*a*Sin[3*Pi*(b + y0)]))}
```

4.15.2.1 case 2

Case $f(x,y)=1$

$$I_i^f = \int_{x=(x_0-a)}^{x=x_0+a} \int_{y=(y_0-b)}^{y=y_0+b} \begin{Bmatrix} \frac{1}{A}((x_0-a)-x)(y_0+b-y) \\ \frac{1}{A}((x_0-a)+x)(y_0+b-y) \\ \frac{1}{A}((x_0-a)+x)(y_0+b+y) \\ \frac{1}{A}((x_0-a)-x)(y_0+b+y) \end{Bmatrix} dxdy$$

This was done using Mathematica

$$I_i^f = \begin{Bmatrix} \frac{a^2b^2}{(a+x_0)(b+y_0)} \\ \frac{ab^2(a+2x_0)}{(a+x_0)(b+y_0)} \\ \frac{ab(a+2x_0)(b+2y_0)}{(a+x_0)(b+y_0)} \\ \frac{a^2b(b+2y_0)}{(a+x_0)(b+y_0)} \end{Bmatrix}$$

```
phi /. {a :> (x0 + a), b :> (y0 + b)}
Integrate[% , {x, x0 - a, x0 + a}, {y, y0 - b, y0 + b}]
{(a^2*b^2)/((a + x0)*(b + y0)),
 (a*b^2*(a + 2*x0))/((a + x0)*(b + y0)),
 (a*b*(a + 2*x0)*(b + 2*y0))/((a + x0)*(b + y0)),
 (a^2*b*(b + 2*y0))/((a + x0)*(b + y0))}
```

Now that the local k_i and local f_i are calculated, the next step is to assemble them to the global stiffness and global force vector. For 1D this process was easy and direct. For higher dimensions we need to make a table of mapping to help in the process. The following diagram shows this mapping table and how it is applied in this example. The mapping table will have as many rows as there are elements, and will have as many columns as the number of nodes per element. Hence in this example it will a matrix of size 9×4. Each row gives the global node number for each local node number of that element. For example, for element 1 the first row will show the global node number for each of the local nodes for the first element.

Local node numbers are always incremented counter clockwise, starting from 1 to 4 in this case.

Now that we have the mapping done, we can now assemble the global stiffness and vector matrix and solve for the unknowns.

To evaluate the force vector, we also need a table of physical coordinates of the center of each element relative to the origin of the global coordinates system, since the force vector integration depends on physical coordinates.

The integration was carried out off-line as shown above, but it was done in terms of physical coordinates of center of the element. The following diagram shows the mapping table of physical coordinates to elements. This uses $(0,0)$ as the origin and the coordinates of the lower left corner of the global coordinate system. Hence the coordinate the top right corner of the global grid will be $(3h, 3h)$ where h is the grid spacing.

Now the solution can start. We loop over each element and fill in the global stiffness matrix using the mapping above, and also evaluate the global force vector using the physical coordinates.

When the global stiffness matrix and the global force vector is filled, we adjust the global stiffness matrix by putting zero in the row numbered i that correspond to the edge node i, and by putting 1 in the diagonal entry (i,i). We also put a zero in the force vector for each node on the boundary conditions since we are using Dirichlet boundary conditions which is zero. When this is done, the system $Ax = f$ is then solved for x where x now is the displacement or value of y at the nodes. The following is an implementation of the above.

```
function matlab_97()
close all; clear all;

number_of_elements=[16 64 256 625];
function_string={'1','xy','3(\cos(4\pi x)+\sin(3 \pi y))'};
function_handles={@f_1,@f_xy,@f_2};

figure_image_file={'matlab_e97_1','matlab_e97_2','matlab_e97_3'};
```

```
for i=1:length(function_handles)
   for j=1:length(number_of_elements)
      plot_file=sprintf('%s_%d',figure_image_file{i},...
                                          number_of_elements(j));
      close all;
      solve_poisson_2D_square(function_handles{i},...
          function_string{i},number_of_elements(j),plot_file);
   end
end
end

%---------------------------------
function k_local=getk(a,b)

k11=(a^2+b^2)/(3*a*b);
k12=a/(6*b)-b/(3*a);
k13=-(a^2+b^2)/(6*a*b);
k14=-a/(3*b)+b/(6*a);
k_local = [k11  k12 k13 k14;
           k12  k11 k14 k13;
           k13  k14 k11 k12;
           k14  k13 k12 k11];
end
%-----------------------------------------------------
%use this for f(x,y)=x*y
function f_local=f_xy(x0,y0,a,b)

 f_local=1/(9*(a + x0)*(b + y0))*...
     [a^2*b^2*(a-3*x0)*(b -3*y0);
      -a*b^2*(a^2 + 3*a*x0 + 6*x0^2)*(b - 3*y0);
      a*b*(a^2 + 3*a*x0 + 6*x0^2)*(b^2 + 3*b*y0 + 6*y0^2);
      -a^2*b*(a - 3*x0)*(b^2 + 3*b*y0 + 6*y0^2)...
     ];
end
%-----------------------------------------------------
%use this for f(x,y)=1
function f_local=f_1(x0,y0,a,b)

f_local=[(a^2*b^2)/((a + x0)*(b + y0));
(a*b^2*(a + 2*x0))/((a + x0)*(b + y0));
  (a*b*(a + 2*x0)*(b + 2*y0))/((a + x0)*(b + y0));
```

```
  (a^2*b*(b + 2*y0))/((a + x0)*(b + y0))];

end
%-----------------------------------------------------
%use this for f(x,y)= 3(cos(4 pi x)+sin(3 pi y))
function f_local=f_2(x0,y0,a,b)
f_local=[(1/(96*pi^2*(a + x0)*(b + y0)))*...
           (9*b^2*cos(4*pi*(-a + x0)) -...
        9*b^2*cos(4*pi*(a + x0)) + ...
             8*a*(12*a*b*pi*cos(3*pi*(b - y0)) -...
             9*b^2*pi*sin(4*pi*(-a + x0)) - ...
                  2*a*(sin(3*pi*(b - y0)) +...
                  sin(3*pi*(b + y0)))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (-9*b^2*cos(4*pi*(-a + x0)) + 9*b^2*cos(4*pi*(a + x0)) +...
     8*(12*a*b*pi*(a + 2*x0)*cos(3*pi*(b - y0)) -...
     9*b^2*pi*x0*sin(4*pi*(-a + x0)) + ...
     9*a*b^2*pi*sin(4*pi*(a + x0)) +...
     9*b^2*pi*x0*sin(4*pi*(a + x0)) - 2*a^2*sin(3*pi*(b - y0)) -...
     4*a*x0*sin(3*pi*(b - y0)) - 2*a^2*sin(3*pi*(b + y0)) -...
     4*a*x0*sin(3*pi*(b + y0))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (-9*b*(b + 2*y0)*cos(4*pi*(-a + x0)) + ...
     9*b*(b + 2*y0)*cos(4*pi*(a + x0)) -...
     72*b*pi*x0*(b + 2*y0)*sin(4*pi*(-a + x0)) + ...
     72*b*pi*(a + x0)*(b + 2*y0)*...
     sin(4*pi*(a + x0)) + 12*(a + x0)^2*...
     (6*pi*y0*cos(3*pi*(b - y0)) -...
     6*pi*(b + y0)*cos(3*pi*(b + y0)) + sin(3*pi*(b - y0)) +...
     sin(3*pi*(b + y0))) - 4*(-a + x0)*(a + 3*x0)*...
    (6*pi*y0*cos(3*pi*(b - y0)) -...
     6*pi*(b + y0)*cos(3*pi*(b + y0)) + sin(3*pi*(b - y0)) +...
             sin(3*pi*(b + y0))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (9*b*(b + 2*y0)*cos(4*pi*(-a + x0)) - ...
        9*b*(b + 2*y0)*cos(4*pi*(a + x0)) +...
        8*a*(12*a*pi*y0*cos(3*pi*(b - y0)) - ...
           12*a*pi*(b + y0)*cos(3*pi*(b + y0)) -...
             9*b^2*pi*sin(4*pi*(-a + x0)) - ...
             18*b*pi*y0*sin(4*pi*(-a + x0)) +...
             2*a*sin(3*pi*(b - y0)) + 2*a*sin(3*pi*(b + y0))))];
```

```
end
%-----------------------------------------------------
function solve_poisson_2D_square(fh,rhs,M,plot_file)
%2a is width of element
%2b is height of element
%fh is function which evaluate the force vector
%rhs is string which is the f(x,y), for plotting only
%M=total number of elements
%k=local stiffness matrix
L  = 1; %length of grid
N  = 4; %number of nodes per element
M1 = sqrt(M); %number of elements per row
a=(L/M1)/2;
b=(L/M1)/2;
k_local=getk(a,b);
N1 = M1+1; %number of nodes per edge
dof=4;
nodes = N1^2; %total number of nodes
kk    = zeros(nodes); %global stiffness vector
f     = zeros(nodes,1); %global force vector
mtb   = generate_coordinates_table(M1);
fmtb  = generate_physical_coordinates_table(M1,a,b);

for i = 1:M
    x0 = fmtb(i,1); y0 = fmtb(i,2);
    %call to load different force vector, pre-computed offline
    f_local=fh(x0,y0,a,b);

    %assemble force vecotr and global stiffness matrix
    for ii = 1:N
        f(mtb(i,ii)) = f(mtb(i,ii)) + f_local(ii);
        for jj = 1:N
            kk(mtb(i,ii),mtb(i,jj)) = kk(mtb(i,ii),mtb(i,jj))+...
                k_local(ii,jj);
        end
    end
end

%fix the global stiffness matrix and force vector for B.C.
edge_nodes=[1:N1 (M1+2):(M1+1):nodes-N1 ...
```

```
                    (2*M1+2):(M1+1):(nodes-N1) nodes-N1:nodes];
for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,:)=0;
    f(z)=0;
end

for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,z)=1;
end

y=kk\f; %solve
Z=reshape(y,N1,N1);
[X,Y] = meshgrid(0:2*a:1,0:2*b:1);
h=surf(X,Y,Z);
shading interp
set(h','edgecolor','k');
set(gcf,'defaulttextinterpreter','latex');
plot_title=sprintf('$\\nabla^2 u=%s$, number of elements $%d$',rhs,M);
title(plot_title,'fontweight','bold','fontsize',14);

print(gcf, '-dpdf', '-r300', ['images/',plot_file]);
print(gcf, '-dpng', '-r300', ['images/',plot_file]);

figure;
[C,h] = contourf(X,Y,Z);
clabel(C,h)
set(gcf,'defaulttextinterpreter','latex');
title(plot_title,'fontweight','bold','fontsize',14);
colormap cool
print(gcf, '-dpdf', '-r300', ['images/',[plot_file,'_c']]);
print(gcf, '-dpng', '-r300', ['images/',[plot_file,'_c']]);

end
%-----------------------------------------------------
function mtb=generate_coordinates_table(M)
%M=number_of_elements_per_row
mtb=zeros(M^2,4);
for i=1:M
    for j=1:M
```

```
        mtb(j+M*(i-1),:) =[j j+1 j+(M+2) j+(M+1)]+(M+1)*(i-1);
    end
end
end
%-----------------------------------------------------
function fmtb=generate_physical_coordinates_table(M,a,b)
%2a is width of element
%2b is height of element
%M=number_of_elements_per_row
fmtb=zeros(M^2,2);
for i=1:M
    for j=1:M
        fmtb(j+M*(i-1),:) = [(2*j-1)*a (2*i-1)*b];
    end
end
end
```

Here is the output for the three force functions, for different number of elements.

4.15.3 $f(x, y) = 1$

4.15.4 $f(x, y) = xy$

4.15.5 case 3

Case 3: $f(x, y) = 3(\cos(4\pi x) + \sin(3\pi y))$

4.15.6 Solving for reactangle grid

A small modification is now made to allow one to specify different width and height for the domain.

```
function matlab_97_2()
close all; clear all;

%number of elements per unit length, width, height
number_of_elements=[4 1 1;
                    8 1 2;
                    16 1 4;
```

```
                    25 1 6];
function_string={'1','xy','3(\cos(4\pi x)+\sin(3 \pi y))'};
function_handles={@f_1,@f_xy,@f_2};

figure_image_file={'matlab_e97_rectangle_1',...
           'matlab_e97_rectangle_2','matlab_e97_rectangle_3'};

for i=1:length(function_handles)
   for j=1:size(number_of_elements,1)
      plot_file=sprintf('%s_%d',figure_image_file{i},...
          number_of_elements(j,1)*number_of_elements(j,2)*...
                                      number_of_elements(j,3));
      close all;
      solve_poisson_2D_square(function_handles{i},...
          function_string{i},number_of_elements(j,1),...
          number_of_elements(j,2),number_of_elements(j,3),...
          plot_file);
   end
end
end

%---------------------------------
function k_local=getk(a,b)

k11=(a^2+b^2)/(3*a*b);
k12=a/(6*b)-b/(3*a);
k13=-(a^2+b^2)/(6*a*b);
k14=-a/(3*b)+b/(6*a);
k_local = [k11  k12 k13 k14;
           k12  k11 k14 k13;
           k13  k14 k11 k12;
           k14  k13 k12 k11];
end
%-----------------------------------------------------
%use this for f(x,y)=x*y
function f_local=f_xy(x0,y0,a,b)

 f_local=1/(9*(a + x0)*(b + y0))*...
     [a^2*b^2*(a-3*x0)*(b -3*y0);
      -a*b^2*(a^2 + 3*a*x0 + 6*x0^2)*(b - 3*y0);
      a*b*(a^2 + 3*a*x0 + 6*x0^2)*(b^2 + 3*b*y0 + 6*y0^2);
```

```
      -a^2*b*(a - 3*x0)*(b^2 + 3*b*y0 + 6*y0^2)...
     ];
end
%-----------------------------------------------------
%use this for f(x,y)=1
function f_local=f_1(x0,y0,a,b)

f_local=[(a^2*b^2)/((a + x0)*(b + y0));
(a*b^2*(a + 2*x0))/((a + x0)*(b + y0));
  (a*b*(a + 2*x0)*(b + 2*y0))/((a + x0)*(b + y0));
  (a^2*b*(b + 2*y0))/((a + x0)*(b + y0))];

end
%-----------------------------------------------------
%use this for f(x,y)= 3(cos(4 pi x)+sin(3 pi y))
function f_local=f_2(x0,y0,a,b)
f_local=[(1/(96*pi^2*(a + x0)*(b + y0)))*...
  (9*b^2*cos(4*pi*(-a + x0)) -...
  9*b^2*cos(4*pi*(a + x0)) + 8*a*(12*a*b*pi*cos(3*pi*(b - y0)) -...
  9*b^2*pi*sin(4*pi*(-a + x0)) - 2*a*(sin(3*pi*(b - y0)) +...
                 sin(3*pi*(b + y0)))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (-9*b^2*cos(4*pi*(-a + x0)) + 9*b^2*cos(4*pi*(a + x0)) +...
        8*(12*a*b*pi*(a + 2*x0)*cos(3*pi*(b - y0)) -...
         9*b^2*pi*x0*sin(4*pi*(-a + x0)) + ...
         9*a*b^2*pi*sin(4*pi*(a + x0)) +...
         9*b^2*pi*x0*sin(4*pi*(a + x0)) - 2*a^2*sin(3*pi*(b - y0)) -...
         4*a*x0*sin(3*pi*(b - y0)) - 2*a^2*sin(3*pi*(b + y0)) -...
         4*a*x0*sin(3*pi*(b + y0))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (-9*b*(b + 2*y0)*cos(4*pi*(-a + x0)) + 9*b*(b + 2*y0)*...
        cos(4*pi*(a + x0)) -...
        72*b*pi*x0*(b + 2*y0)*sin(4*pi*(-a + x0)) + ...
        72*b*pi*(a + x0)*(b + 2*y0)*...
          sin(4*pi*(a + x0)) + 12*(a + x0)^2*(6*pi*y0*...
             cos(3*pi*(b - y0)) -...
             6*pi*(b + y0)*cos(3*pi*(b + y0)) + ...
             sin(3*pi*(b - y0)) +...
             sin(3*pi*(b + y0))) - 4*(-a + x0)*(a + 3*x0)*...
            (6*pi*y0*cos(3*pi*(b - y0)) -...
             6*pi*(b + y0)*cos(3*pi*(b + y0)) + ...
```

```
            sin(3*pi*(b - y0)) +...
            sin(3*pi*(b + y0))));
     (1/(96*pi^2*(a + x0)*(b + y0)))*...
     (9*b*(b + 2*y0)*cos(4*pi*(-a + x0)) - 9*b*(b + 2*y0)*...
     cos(4*pi*(a + x0)) +...
     8*a*(12*a*pi*y0*cos(3*pi*(b - y0)) - 12*a*pi*(b + y0)*...
     cos(3*pi*(b + y0)) -...
     9*b^2*pi*sin(4*pi*(-a + x0)) - 18*b*pi*y0*sin(4*pi*(-a + x0)) +...
     2*a*sin(3*pi*(b - y0)) + 2*a*sin(3*pi*(b + y0))))];

end
%-----------------------------------------------------
function solve_poisson_2D_square(fh,rhs,M,Lx,Ly,plot_file)
%fh is function which evaluate the force vector
%rhs is string which is the f(x,y), for plotting only
%M=number of element per unit length
%Lx=length in x-direction
%Ly=length in y-direction
N  = 4; %number of nodes per element
Mx=M*Lx;
My=M*Ly;
a=(1/M)/2;
b=a;
k_local=getk(a,b);
dof=4;

nodes = (Mx+1)*(My+1); %total number of nodes
kk    = zeros(nodes); %global stiffness vector
f     = zeros(nodes,1); %global force vector
mtb   = generate_coordinates_table(Mx,My);
fmtb  = generate_physical_coordinates_table(Mx,My,a,b);

for i = 1:(Mx*My)
    x0 = fmtb(i,1); y0 = fmtb(i,2);
    %call to load different force vector, pre-computed offline
    f_local=fh(x0,y0,a,b);

    %assemble force vecotr and global stiffness matrix
    for ii = 1:N
        f(mtb(i,ii)) = f(mtb(i,ii)) + f_local(ii);
        for jj = 1:N
```

```
            kk(mtb(i,ii),mtb(i,jj)) = kk(mtb(i,ii),mtb(i,jj))+...
                k_local(ii,jj);
        end
    end
end

%fix the global stiffness matrix and force vector for B.C.
edge_nodes=[1:Mx+1 (Mx+2):(Mx+1):nodes-...
           Mx (2*Mx+2):(Mx+1):(nodes-(Mx+1)) nodes-(Mx-1):nodes];
for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,:)=0;
    f(z)=0;
end

for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,z)=1;
end

y=kk\f; %solve
Z=reshape(y,Mx+1,My+1);
[X,Y] = meshgrid(0:2*b:Ly,0:2*a:Lx);
h=surf(X,Y,Z);
shading interp
set(h','edgecolor','k');
set(gcf,'defaulttextinterpreter','latex');
plot_title=sprintf('$\\nabla^2 u=%s$, number of elements $%d$',rhs,Mx*My);
title(plot_title,'fontweight','bold','fontsize',14);

print(gcf, '-dpdf', '-r300', ['images/',plot_file]);
print(gcf, '-dpng', '-r300', ['images/',plot_file]);

figure;
[C,h] = contourf(X,Y,Z);
clabel(C,h)
set(gcf,'defaulttextinterpreter','latex');
title(plot_title,'fontweight','bold','fontsize',14);
colormap cool

print(gcf, '-dpdf', '-r300', ['images/',[plot_file,'_c']]);
```

```
print(gcf, '-dpng', '-r300', ['images/',[plot_file,'_c']]);

end
%-----------------------------------------------------
function mtb=generate_coordinates_table(Mx,My)
%Mx= number of element x-wise
%My= number of element y-wise
mtb=zeros(Mx*My,4);
for i=1:My
    for j=1:Mx
        mtb(j+Mx*(i-1),:) =[j j+1 j+(Mx+2) j+(Mx+1)]+(Mx+1)*(i-1);
    end
end
end
%-----------------------------------------------------
function fmtb=generate_physical_coordinates_table(Mx,My,a,b)
%Mx= number of element x-wise
%My= number of element y-wise
%2a is width of element
%2b is height of element
fmtb=zeros(Mx*My,2);
for i=1:My
    for j=1:Mx
        fmtb(j+Mx*(i-1),:) = [(2*j-1)*a (2*i-1)*b];
    end
end
end
```

Here is the output for the three force functions, for different number of elements.

4.15.7 Case 4

Case $f(x, y) = 1$.

4.15.8 Case 5

Case : $f(x, y) = xy$

4.15.9 case 6

Case $f(x, y) = 3(\cos(4\pi x) + \sin(3\pi y))$.

4.16 How to solve Poisson PDE in 2D using finite elements methods using triangle element?

Solve $\nabla^2 u = f(x,y)$ on square using FEM. Assume $u = 0$ on boundaries and solve using $f(x,y) = xy$ and also $f(x,y) = 1$ and $f(x,y) = 3(\cos(4\pi x) + \sin(3\pi y))$

Use Galerkin method and weak form, Using triangle element.

Solution

Using as an example with 9 elements to illustrate the method. The program below can be called with different number of elements.

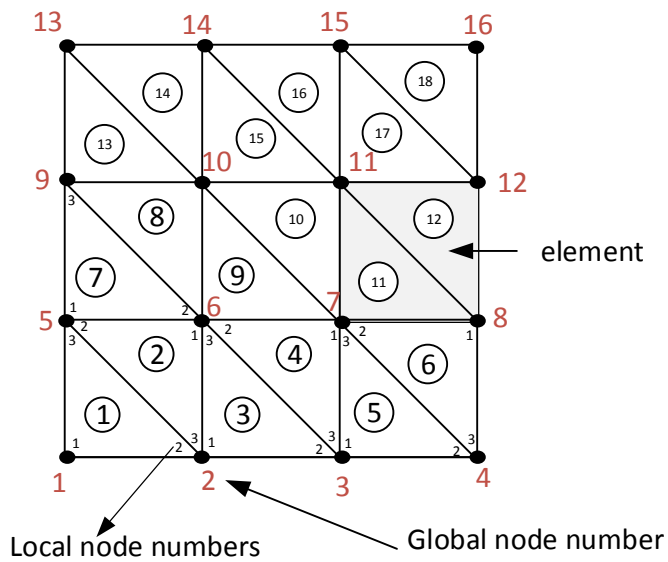


Figure 4.1: node numbering

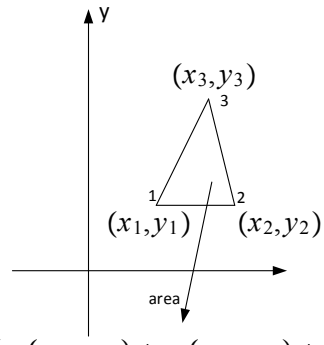
Looking at one element

The trial function is

$$\tilde{u} = c_1 + c_2 x + c_3 y \tag{1}$$

Evaluating the trial function (1) at each corner node of the above element gives

$$\begin{aligned}
\tilde{u}_1 &= c_1 + c_2 x_1 + c_3 y_1 \\
\tilde{u}_2 &= c_1 + c_2 x_2 + c_3 y_2 \\
\tilde{u}_3 &= c_1 + c_2 x_3 + c_3 y_3
\end{aligned}$$



$A = \frac{1}{2}(x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2))$

Figure 4.2: element in physical coordinates

Hence

$$\begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

Therefore

$$\begin{aligned} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} &= \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}^{-1} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} \\ &= \frac{1}{|D|} \begin{pmatrix} x_2y_3 - x_3y_2 & x_3y_1 - x_1y_3 & x_1y_2 - y_1x_2 \\ y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} \end{aligned} \tag{2}$$

Where $|D|$ is the determinant of $\begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}$ which is $|D| = -y_1x_2 + x_3y_1 + x_1y_2 - x_3y_2 - x_1y_3 + x_2y_3$.

This quantity is twice the area of the triangle $A = \frac{1}{2}(x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2))$.

Substituting (2) into (1) in order to find the shape functions gives

$$\begin{aligned}\tilde{u} &= c_1 + c_2 x + c_3 y \\ &= \frac{1}{2A}\left[\left(x_2 y_3 - x_3 y_2\right)\tilde{u}_1 + \left(x_3 y_1 - x_1 y_3\right)\tilde{u}_2 + \left(x_1 y_2 - y_1 x_2\right)\tilde{u}_3\right] \\ &+ \frac{1}{2A}\left[\left(y_2 - y_3\right)\tilde{u}_1 + \left(y_3 - y_1\right)\tilde{u}_2 + \left(y_1 - y_2\right)\tilde{u}_3\right] x \\ &+ \frac{1}{2A}\left[\left(x_3 - x_2\right)\tilde{u}_1 + \left(x_1 - x_3\right)\tilde{u}_2 + \left(x_2 - x_1\right)\tilde{u}_3\right] y\end{aligned}$$

Collecting terms in $\tilde{u}_i$

$$\begin{aligned}\tilde{u} = \tilde{u}_1 &\frac{1}{2A}\left(\left(x_2 y_3 - x_3 y_2\right) + \left(y_2 - y_3\right) x + \left(x_3 - x_2\right) y\right) \\ + \tilde{u}_2 &\frac{1}{2A}\left(\left(x_3 y_1 - x_1 y_3\right) + \left(y_3 - y_1\right) x + \left(x_1 - x_3\right) y\right) \\ + \tilde{u}_3 &\frac{1}{2A}\left(\left(x_1 y_2 - y_1 x_2\right) + \left(y_1 - y_2\right) x + \left(x_2 - x_1\right) y\right)\end{aligned}$$

Hence

$$\begin{aligned}N_1(x,y) &= \frac{1}{2A}\left(\left(x_2 y_3 - x_3 y_2\right) + \left(y_2 - y_3\right) x + \left(x_3 - x_2\right) y\right) \\ N_2(x,y) &= \frac{1}{2A}\left(\left(x_3 y_1 - x_1 y_3\right) + \left(y_3 - y_1\right) x + \left(x_1 - x_3\right) y\right) \\ N_3(x,y) &= \frac{1}{2A}\left(\left(x_1 y_2 - y_1 x_2\right) + \left(y_1 - y_2\right) x + \left(x_2 - x_1\right) y\right)\end{aligned}$$

Therefore (1) can be written as

$$\tilde{u} = N_1(x,y)\,\tilde{u}_1 + N_2(x,y)\,\tilde{u}_2 + N_3(x,y)\,\tilde{u}_3$$

Now that the shape functions are found, the test or weight function $w(x,y)$ can be found. Since $w_i(x,y) = \frac{\partial \tilde{u}}{\partial \tilde{u}_i}$ for $i = 1,2,3$, therefore this results in

$$w(x,y) = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} = \frac{1}{2A}\begin{pmatrix} \left(x_2 y_3 - x_3 y_2\right) + \left(y_2 - y_3\right) x + \left(x_3 - x_2\right) y \\ \left(x_3 y_1 - x_1 y_3\right) + \left(y_3 - y_1\right) x + \left(x_1 - x_3\right) y \\ \left(x_1 y_2 - y_1 x_2\right) + \left(y_1 - y_2\right) x + \left(x_2 - x_1\right) y \end{pmatrix}$$

The weak form formulation now gives

$$I_i = -\int_\Omega \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} d\Omega - \int_\Omega \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} d\Omega + \int_\Gamma w\frac{\partial \tilde{u}}{\partial n} d\Gamma - \int_\Omega \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} f(x,y)\, d\Omega$$

Hence

$$\begin{aligned} I_i = &-\int_\Omega \begin{pmatrix} \frac{\partial N_1(x,y)}{\partial x} \\ \frac{\partial N_2(x,y)}{\partial x} \\ \frac{\partial N_3(x,y)}{\partial x} \end{pmatrix} \begin{Bmatrix} \frac{\partial N_1(x,y)}{\partial x} & \frac{\partial N_2(x,y)}{\partial x} & \frac{\partial N_3(x,y)}{\partial x} \end{Bmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} d\Omega \\ &-\int_\Omega \begin{pmatrix} \frac{\partial N_1(x,y)}{\partial y} \\ \frac{\partial N_2(x,y)}{\partial y} \\ \frac{\partial N_3(x,y)}{\partial y} \end{pmatrix} \begin{Bmatrix} \frac{\partial N_1(x,y)}{\partial y} & \frac{\partial N_2(x,y)}{\partial y} & \frac{\partial N_3(x,y)}{\partial y} \end{Bmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} d\Omega \\ &+\int_\Gamma \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} \frac{\partial \tilde{u}}{\partial n} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} d\Gamma \\ &-\int_\Omega \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} f(x,y)\, d\Omega \end{aligned}$$

Since $w = 0$ on the boundary with Dirichlet conditions, and since there are no natural boundary conditions, the above simplifies to

$$\begin{aligned} I_i = &-\int_\Omega \begin{pmatrix} \frac{\partial N_1(x,y)}{\partial x} \\ \frac{\partial N_2(x,y)}{\partial x} \\ \frac{\partial N_3(x,y)}{\partial x} \end{pmatrix} \begin{Bmatrix} \frac{\partial N_1(x,y)}{\partial x} & \frac{\partial N_2(x,y)}{\partial x} & \frac{\partial N_3(x,y)}{\partial x} \end{Bmatrix} \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} d\Omega \\ &-\int_\Omega \begin{pmatrix} \frac{\partial N_1(x,y)}{\partial y} \\ \frac{\partial N_2(x,y)}{\partial y} \\ \frac{\partial N_3(x,y)}{\partial y} \end{pmatrix} \begin{Bmatrix} \frac{\partial N_1(x,y)}{\partial y} & \frac{\partial N_2(x,y)}{\partial y} & \frac{\partial N_3(x,y)}{\partial y} \end{Bmatrix} d\Omega \begin{Bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{Bmatrix} - \int_\Omega \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} f(x,y)\, d\Omega \end{aligned}$$

Now each integral is calculated for the triangle element. First the derivatives are found

$$\begin{pmatrix} \frac{\partial N_1(x,y)}{\partial x} \\ \frac{\partial N_2(x,y)}{\partial x} \\ \frac{\partial N_3(x,y)}{\partial x} \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} (y_2 - y_3) \\ (y_3 - y_1) \\ (y_1 - y_2) \end{pmatrix}$$

and

$$\begin{pmatrix} \frac{\partial N_1(x,y)}{\partial y} \\ \frac{\partial N_2(x,y)}{\partial y} \\ \frac{\partial N_3(x,y)}{\partial y} \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} (x_3 - x_2) \\ (x_1 - x_3) \\ (x_2 - x_1) \end{pmatrix}$$

Hence

$$\begin{pmatrix} \frac{\partial N_1(x,y)}{\partial x} \\ \frac{\partial N_2(x,y)}{\partial x} \\ \frac{\partial N_3(x,y)}{\partial x} \end{pmatrix} \left\{ \frac{\partial N_1(x,y)}{\partial x} \quad \frac{\partial N_2(x,y)}{\partial x} \quad \frac{\partial N_3(x,y)}{\partial x} \right\} = \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x}$$

$$= \frac{1}{(2A)^2} \begin{pmatrix} (y_2 - y_3)^2 & (y_2 - y_3)(y_3 - y_1) & (y_2 - y_3)(y_1 - y_2 \\ (y_3 - y_1)(y_2 - y_3) & (y_3 - y_1)^2 & (y_3 - y_1)(y_1 - y_2 \\ (y_1 - y_2)(y_2 - y_3) & (y_1 - y_2)(y_3 - y_1) & (y_1 - y_2)^2 \end{pmatrix}$$

And

$$\begin{pmatrix} \frac{\partial N_1(x,y)}{\partial y} \\ \frac{\partial N_2(x,y)}{\partial y} \\ \frac{\partial N_3(x,y)}{\partial y} \end{pmatrix} \left\{ \frac{\partial N_1(x,y)}{\partial y} \quad \frac{\partial N_2(x,y)}{\partial y} \quad \frac{\partial N_3(x,y)}{\partial y} \right\} = \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y}$$

$$= \frac{1}{(2A)^2} \begin{pmatrix} (x_3 - x_2)^2 & (x_3 - x_2)(x_1 - x_3) & (x_3 - x_2)(x_2 - x_1) \\ (x_1 - x_3)(x_3 - x_2) & (x_1 - x_3)^2 & (x_1 - x_3)(x_2 - x_1) \\ (x_2 - x_1)(x_3 - x_2) & (x_2 - x_1)(x_1 - x_3) & (x_2 - x_1)^2 \end{pmatrix}$$

We see the above terms do not depend on x nor on y. Hence the next integration is done over the area of the triangle as follow

$$\int_{\Omega} \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega = \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}\int_{\Omega} d\Omega$$

But $\int_{\Omega} d\Omega$ is the area of the triangle. Hence

$$\int_{\Omega} \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega = \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} A$$

Replacing $\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}$ by the expression we found for it above, gives

$$\begin{aligned}\int_{\Omega} \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega &= \frac{A}{(2A)^2}\begin{pmatrix}(y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2)\\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2)\\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}\\ &= \frac{1}{4A}\begin{pmatrix}(y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2)\\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2)\\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}\end{aligned} \tag{3}$$

Now we do the same for the second integral

$$\int_{\Omega} \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega = \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}\int_{\Omega} d\Omega$$

But $\int_{\Omega} d\Omega$ is the area of the triangle. Hence

$$\int_{\Omega} \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega = \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} A$$

Replacing $\frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y}$ by the expression we found for it above, gives

$$\int_{\Omega} \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} d\Omega = \frac{A}{(2A)^2}\begin{pmatrix}(x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1)\\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1)\\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}$$

$$= \frac{1}{4A}\begin{pmatrix}(x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1)\\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1)\\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix} \tag{4}$$

Adding (3) and (4) to obtain the local stiffness matrix

$$k_i\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}_i = \frac{1}{4A}\begin{pmatrix}(y_2-y_3)^2 & (y_2-y_3)(y_3-y_1) & (y_2-y_3)(y_1-y_2)\\ (y_3-y_1)(y_2-y_3) & (y_3-y_1)^2 & (y_3-y_1)(y_1-y_2)\\ (y_1-y_2)(y_2-y_3) & (y_1-y_2)(y_3-y_1) & (y_1-y_2)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}$$

$$+ \frac{1}{4A}\begin{pmatrix}(x_3-x_2)^2 & (x_3-x_2)(x_1-x_3) & (x_3-x_2)(x_2-x_1)\\ (x_1-x_3)(x_3-x_2) & (x_1-x_3)^2 & (x_1-x_3)(x_2-x_1)\\ (x_2-x_1)(x_3-x_2) & (x_2-x_1)(x_1-x_3) & (x_2-x_1)^2\end{pmatrix}\begin{Bmatrix}\tilde{u}_1\\ \tilde{u}_2\\ \tilde{u}_3\end{Bmatrix}$$

or

$$k_i = \frac{1}{4A}\begin{pmatrix}(y_2-y_3)^2+(x_3-x_2)^2 & (y_2-y_3)(y_3-y_1)+(x_3-x_2)(x_1-x_3) & (y_2-y_3)(y_1-y_2)+(x_3- \\ (y_3-y_1)(y_2-y_3)+(x_1-x_3)(x_3-x_2) & (y_3-y_1)^2+(x_1-x_3)^2 & (y_3-y_1)(y_1-y_2)+(x_1- \\ (y_1-y_2)(y_2-y_3)+(x_2-x_1)(x_3-x_2) & (y_1-y_2)(y_3-y_1)+(x_2-x_1)(x_1-x_3) & (y_1-y_2)^2+(x_2- \end{pmatrix}$$

And the final integral (the force vector) depends on the nature of $f(x,y)$. For $f(x,y)=1$ we obtain

$$\int_{\Omega}\begin{pmatrix}N_1(x,y)\\ N_2(x,y)\\ N_3(x,y)\end{pmatrix} f(x,y)\,d\Omega = \int_{\Omega}\frac{1}{2A}\begin{pmatrix}(x_2y_3-x_3y_2)+(y_2-y_3)\,x+(x_3-x_2)\,y\\ (x_3y_1-x_1y_3)+(y_3-y_1)\,x+(x_1-x_3)\,y\\ (x_1y_2-y_1x_2)+(y_1-y_2)\,x+(x_2-x_1)\,y\end{pmatrix} dydx$$

And for $f(x,y)=xy$ similarly

$$\int_{\Omega} \begin{pmatrix} N_1(x,y) \\ N_2(x,y) \\ N_3(x,y) \end{pmatrix} f(x,y)\, d\Omega = \int_{\Omega} \frac{1}{2A} \begin{pmatrix} (x_2y_3 - x_3y_2) + (y_2 - y_3)\, x + (x_3 - x_2)\, y \\ (x_3y_1 - x_1y_3) + (y_3 - y_1)\, x + (x_1 - x_3)\, y \\ (x_1y_2 - y_1x_2) + (y_1 - y_2)\, x + (x_2 - x_1)\, y \end{pmatrix} xydydx$$

And similarly for other $f(x,y)$. These were all integrated off-line and the resulting expression evaluated during the run of the program. These expression are all functions of the node coordinates $(x_1, y_1, x_2, y_2, x_3, y_3)$. No actual integration is done in the code, but only evaluation of the integration result obtained. The integration was done using Mathematica. These are the results for the three functions of interest. The integration was done on two triangles. The odd numbered ones and the even numbered ones due to the node numbering difference. For odd numbered triangles, the integration limits are $\int_{x_1}^{x_2} \int_{y_1}^{y_3-x} dydx$ while on the even numbered elements the limits are $\int_{x_2}^{x_1} \int_{y_1}^{y_2-x} dydx$ as shown in this diagram

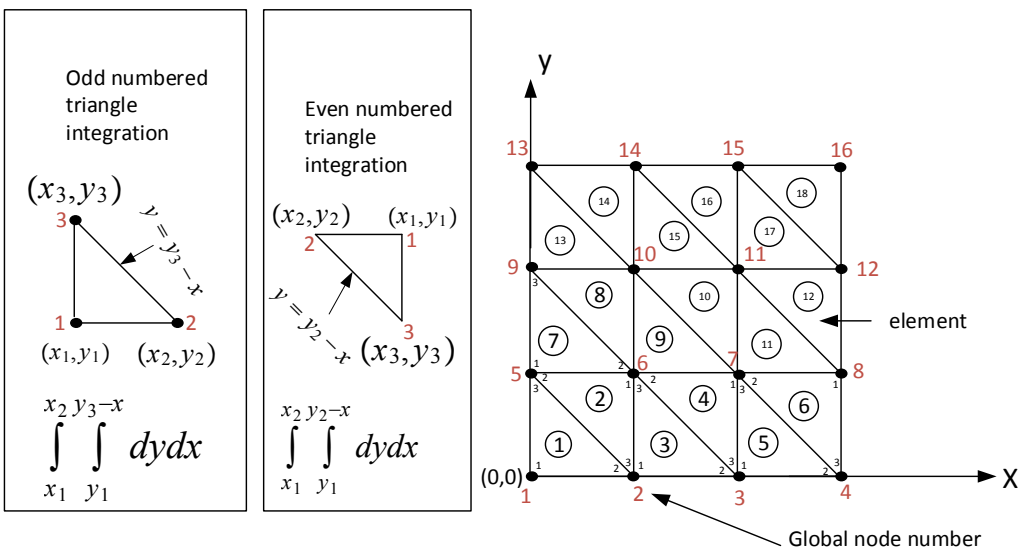


Figure 4.3: Integrating over elements

Note that this is not how integration is normally done in actual finite element methods. Using Gaussian quadrature method is the recommended way. This method was done here for learning and to compare.

4.16.1 case 1

Case $f(x, y) = 1$. For odd numbered elements

```
n1 = (1/(2*area))*((x2*y3 - x3*y2) + (y2 - y3)*x + (x3 - x2)*y);
n2 = (1/(2*area))*((x2*y1 - x1*y3) + (y3 - y1)*x + (x1 - x3)*y);
n3 = (1/(2*area))*((x1*y2 - y1*x2) + (y1 - y2)*x + (x2 - x1)*y);
w = {{n1}, {n2}, {n3}};
Integrate[w, {x, x1, x2}, {y, y1, y3 - x}]

Out[11]= {{(1/(12*area))*((x1 - x2)*(x2^3 + x1^2*(x2 - x3 + 2*y2 - 2*y3) +
          3*x3*(y1 - y3)*(y1 - 2*y2 + y3) - x2^2*(x3 - 2*y2 + 2*y3) -
          3*x2*(y1^2 + x3*y2 - x3*y3 + y2*y3 - y1*(y2 + y3)) + x1*
          (x2^2 - 3*(y2 - y3)*(x3 - y1 + y3) - x2*(x3 - 2*y2 + 2*y3))))},
  {-(((x1 - x2)*(x1^3 + x1^2*(x2 - x3 + 2*y1 - 2*y3) - x2^2*(x3 + y1 + 2*y3) +
          3*x3*(y1^2 - y3^2) + 3*x2*(-y1^2 + y3*(x3 + y3)) +
       x1*(x2^2 + 3*x3*y3 - x2*(x3 + y1 + 2*y3))))/(12*area))},
  {((x1 - x2)^2*(x1^2 + x2^2 + x1*(x2 + 2*y1 + y2 - 3*y3) + x2*(y1 + 2*y2 - 3*y3) +
       3*(y1 - y3)*(y2 - y3)))/(12*area)}}
```

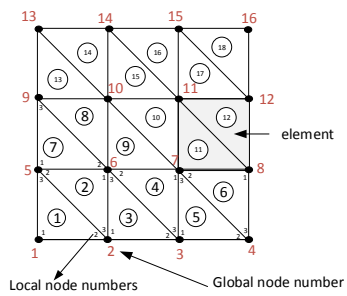
For even numbered elements

```
Integrate[w, {x, x1, x2}, {y, y1, y2 - x}]
{{((x1 - x2)*(x2^3 + 3*x3*(y1 - y2)^2 + x1*(x2^2 + 3*(y1 - y2)*(y2 - y3) -
   x2*(x3 + y2 - y3)) + x1^2*(x2 - x3 + 2*y2 - 2*y3) -
   3*x2*(y1 - y2)*(y1 - y3) - x2^2*(x3 + y2 - y3)))/(12*area)},
   {-((1/(12*area))*((x1 - x2)*(x1^3 - 3*x3*(y1 - y2)^2 - 3*x2*(y1 - y2)*
     (x3 - y1 + y3) + x1^2*(x2 - x3 + 2*y1 - 3*y2 + y3) -
   x2^2*(x3 - 2*y1 + 2*y3) + x1*(x2^2 - 3*(y1 - y2)*(x3 + y2 - y3) +
   x2*(-x3 + 2*y1 - 3*y2 + y3)))))},
 {((x1 - x2)^2*(x1^2 + x1*(x2 + 2*y1 - 2*y2) + x2*(x2 + y1 - y2)))/(12*area)}
}
```

4.16.2 Program output

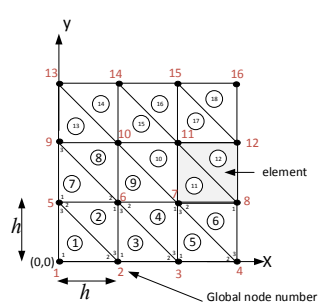
This is an implementation of the above for $f(x,y)=1$. The coordinates mapping was first carried out in order to map each elements local node numbers to the global node numbering so we know where to add each element stiffness matrix to the global stiffness matrix. Also we need to build the mapping of each element to the physical coordinates of its local node number 1 to use for calculating the force vector since that depends on physical location of each element.

$$\underbrace{\begin{pmatrix} 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\\16\\17\\18 \end{pmatrix}}_{\text{Element numbers}} = \overbrace{\underbrace{\begin{pmatrix} 1 & 2 & 5\\ 6 & 5 & 2\\ 2 & 3 & 6\\ 7 & 6 & 3\\ 3 & 4 & 7\\ 8 & 7 & 4\\ 5 & 6 & 9\\ 10 & 9 & 6\\ 6 & 7 & 10\\ 11 & 10 & 7\\ 7 & 8 & 11\\ 12 & 11 & 8\\ 9 & 10 & 13\\ 14 & 13 & 10\\ 10 & 11 & 14\\ 15 & 14 & 11\\ 11 & 12 & 15\\ 16 & 15 & 12 \end{pmatrix}}_{\text{Global node numbers mapping}}}^{(1,2,3)}$$



The mapping of physical coordinate location relative to the global frame of reference is shown below

$$\underbrace{\begin{pmatrix} 1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\\16\\17\\18 \end{pmatrix}}_{\text{Element number}} \Rightarrow \overbrace{\underbrace{\begin{pmatrix} 0 & 0 & h & 0 & 0 & h\\ h & h & 0 & h & h & 0\\ h & 0 & 2h & 0 & h & h\\ 2h & h & h & h & 2h & 0\\ 2h & 0 & 3h & 0 & 2h & h\\ 3h & h & 2h & h & 3h & 0\\ 0 & h & h & h & 0 & 2h\\ h & 2h & 0 & 2h & h & h\\ h & h & 2h & h & h & 2h\\ 2h & 2h & h & 2h & 2h & h\\ 2h & h & 3h & h & 2h & 2h\\ 3h & 2h & 2h & 2h & 3h & h\\ 0 & 2h & h & 2h & 0 & 3h\\ h & 3h & 0 & 3h & h & 2h\\ h & 2h & 2h & 2h & h & 3h\\ 2h & 3h & h & 3h & 2h & 2h\\ 2h & 2h & 3h & 2h & 2h & 3h\\ 3h & 3h & 2h & 3h & 3h & 2h \end{pmatrix}}_{\text{(x,y) coordinates of nodes 1,2,3 of each element}}}^{x_1,y_1,x_2,y_2,x_3,y_3}$$



```
function matlab_98()
close all; clear all;
number_of_elements=[16*2 64*2 256*2 625*2];
function_string={'1'};
%function_handles={@f_1,@f_xy,@f_2};%something wrong with the other
%cases, need to work more on it.
function_handles={@f_1};
figure_image_file={'matlab_e98_1'};

for i=1:length(function_handles)
   for j=1:length(number_of_elements)
      plot_file=sprintf('%s_%d',figure_image_file{i},...
                                          number_of_elements(j));
      close all;
      solve_poisson_2D_square(function_handles{i},...
          function_string{i},number_of_elements(j),plot_file);
   end
end
end

%----------------------------------
function k_local=getk(area,x1,y1,x2,y2,x3,y3)

k11=(y2-y3)^2+(x3-x2)^2;
k12=(y2-y3)*(y3-y1)+(x3-x2)*(x1-x3);
k13=(y2-y3)*(y1-y2)+(x3-x2)*(x2-x1);
k22=(y3-y1)^2+(x1-x3)^2;
k23=(y3-y1)*(y1-y2)+(x1-x3)*(x2-x1);
k33=(y1-y2)^2+(x2-x1)^2;
k_local = [k11  k12 k13;
           k12  k22 k23;
           k13  k23 k33];

k_local = k_local/(4*area);
end
%-----------------------------------------------------
%use this for f(x,y)=1
function f_local=f_1(area,x1,y1,x2,y2,x3,y3,ele)

if mod(ele,2)
  f_local=[(1/(12*area))*((x1 - x2)*(x2^3 + x1^2*...
```

```
   (x2 - x3 + 2*y2 - 2*y3) +...
   3*x3*(y1 - y3)*(y1 - 2*y2 + y3) - x2^2*(x3 - 2*y2 + 2*y3) -...
   3*x2*(y1^2 + x3*y2 - x3*y3 + y2*y3 - y1*(y2 + y3)) + x1*(x2^2 -...
   3*(y2 - y3)*(x3 - y1 + y3) - x2*(x3 - 2*y2 + 2*y3))));
  -((1/(12*area))*((x1 - x2)*(x1^3 + x1^2*(x2 - x3 + 2*y1 - 2*y3) -...
  3*x3*(y1 - y3)^2 - 3*x2*(y1 - y3)*(x3 - y1 + y3) -...
       x2^2*(x3 - 2*y1 + 2*y3) + x1*(x2^2 + 3*x3*(-y1 + y3) -...
       x2*(x3 - 2*y1 + 2*y3)))));
  ((x1 - x2)^2*(x1^2 + x2^2 + x1*(x2 + 2*y1 + y2 - 3*y3) +...
  x2*(y1 + 2*y2 - 3*y3) + 3*(y1 - y3)*(y2 - y3)))/(12*area)];
else
  f_local=[((x1 - x2)*(x2^3 + 3*x3*(y1 - y2)^2 + x1*(x2^2 +...
      3*(y1 - y2)*(y2 - y3) - x2*(x3 + y2 - y3)) + x1^2*...
      (x2 - x3 + 2*y2 - 2*y3) -...
      3*x2*(y1 - y2)*(y1 - y3) - x2^2*(x3 + y2 - y3)))/(12*area);
  -((1/(12*area))*((x1 - x2)*(x1^3 - 3*x3*(y1 - y2)^2 -...
  3*x2*(y1 - y2)*(x3 - y1 + y3) + x1^2*...
       (x2 - x3 + 2*y1 - 3*y2 + y3) -...
       x2^2*(x3 - 2*y1 + 2*y3) + x1*(x2^2 - 3*(y1 - y2)*(x3 + y2 - y3)...
       + x2*(-x3 + 2*y1 - 3*y2 + y3)))));
  ((x1 - x2)^2*(x1^2 + x1*(x2 + 2*y1 - 2*y2) + ...
        x2*(x2 + y1 - y2)))/(12*area)];

end

end
%-----------------------------------------------------
function solve_poisson_2D_square(fh,rhs,M,plot_file)
%fh is function which evaluate the force vector
%rhs is string which is the f(x,y), for plotting only
%M=total number of elements
L  = 1; %length of grid
N  = 3; %number of nodes per element
h=(L/sqrt(M/2));
%area is same for each element, otherwise use
%area = 0.5*(x1*(y2-y3)+x2*(y3-y1)+x3*(y1-y2));
area= (1/2)*h^2;

N1 = sqrt(M/2)+1; %number of nodes per edge
dof=3;
nodes = N1^2; %total number of nodes
```

```
kk    = zeros(nodes); %global stiffness vector
f     = zeros(nodes,1); %global force vector
mtb   = generate_coordinates_table(M);
fmtb  = generate_physical_coordinates_table(M,h);

for i = 1:M
    x1 = fmtb(i,1); y1 = fmtb(i,2);
    x2 = fmtb(i,3); y2 = fmtb(i,4);
    x3 = fmtb(i,5); y3 = fmtb(i,6);
    k_local=getk(area,x1,y1,x2,y2,x3,y3);
    %call to load different force vector, pre-computed offline
    f_local=fh(area,x1,y1,x2,y2,x3,y3,i);

    for ii = 1:N %assemble force vector and global stiffness matrix
        f(mtb(i,ii)) = f(mtb(i,ii)) + f_local(ii);
        for jj = 1:N
            kk(mtb(i,ii),mtb(i,jj)) = kk(mtb(i,ii),mtb(i,jj))+...
                k_local(ii,jj);
        end
    end
end

%fix the global stiffness matrix and force vector for B.C.
edge_nodes=[1:N1 (N1+1):N1:nodes-N1+1 ...
                            (2*N1):N1:nodes nodes-N1+2:nodes-1];
for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,:)=0;
    f(z)=0;
end

for i=1:length(edge_nodes)
    z=edge_nodes(i);
    kk(z,z)=1;
end

y=kk\f; %solve
Z=reshape(y,N1,N1);
[X,Y] = meshgrid(0:h:1,0:h:1);
h=surf(X,Y,Z);
shading interp
```

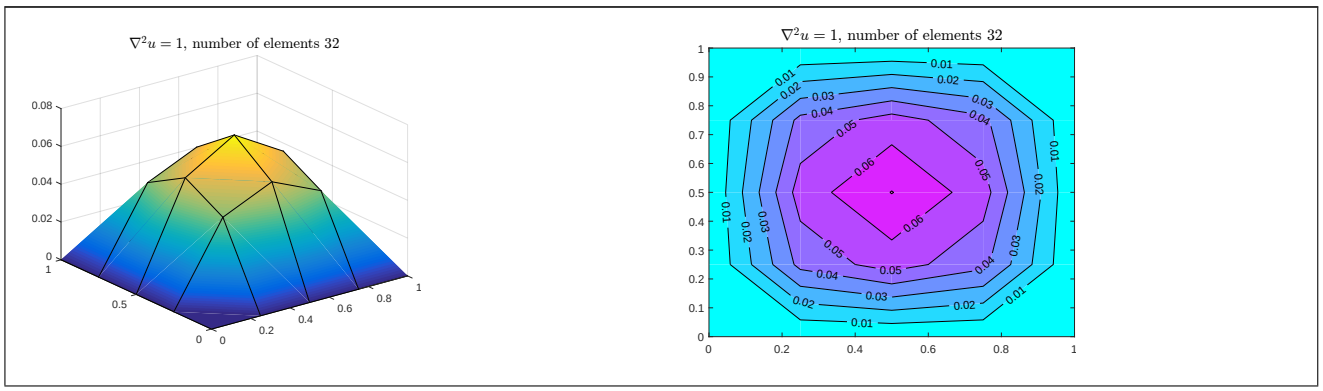
```
set(h','edgecolor','k');
set(gcf,'defaulttextinterpreter','latex');
plot_title=sprintf('$\\nabla^2 u=%s$, number of elements $%d$',rhs,M);
title(plot_title,'fontweight','bold','fontsize',14);

print(gcf, '-dpdf', '-r600', ['images/',plot_file]);
print(gcf, '-dpng', '-r300', ['images/',plot_file]);

figure;
[C,h] = contourf(X,Y,Z);
clabel(C,h)
set(gcf,'defaulttextinterpreter','latex');
title(plot_title,'fontweight','bold','fontsize',14);
colormap cool

print(gcf, '-dpdf', '-r300', ['images/',plot_file,'_c']);
print(gcf, '-dpng', '-r300', ['images/',plot_file,'_c']);

end
%-----------------------------------------------------
function A=generate_coordinates_table(M)
%M=number_of_elements
A=zeros(M,3);
n=2*sqrt(M/2); %number of elements per row

for i=1:n/2
    for j=1:n
        ele=j+n*(i-1);
        if j==1
           A(ele,:)=[(n/2+1)*(i-1)+1 (n/2+1)*(i-1)+2 (n/2+1)*i+1];
        else
            if mod(j,2)
                A(ele,:) =[A(ele-1,3) A(ele-1,3)+1 A(ele-1,1)];
            else
                A(ele,:) =[A(ele-1,3)+1 A(ele-1,3) A(ele-1,2)];
            end
        end
    end
end
end
%-----------------------------------------------------
```

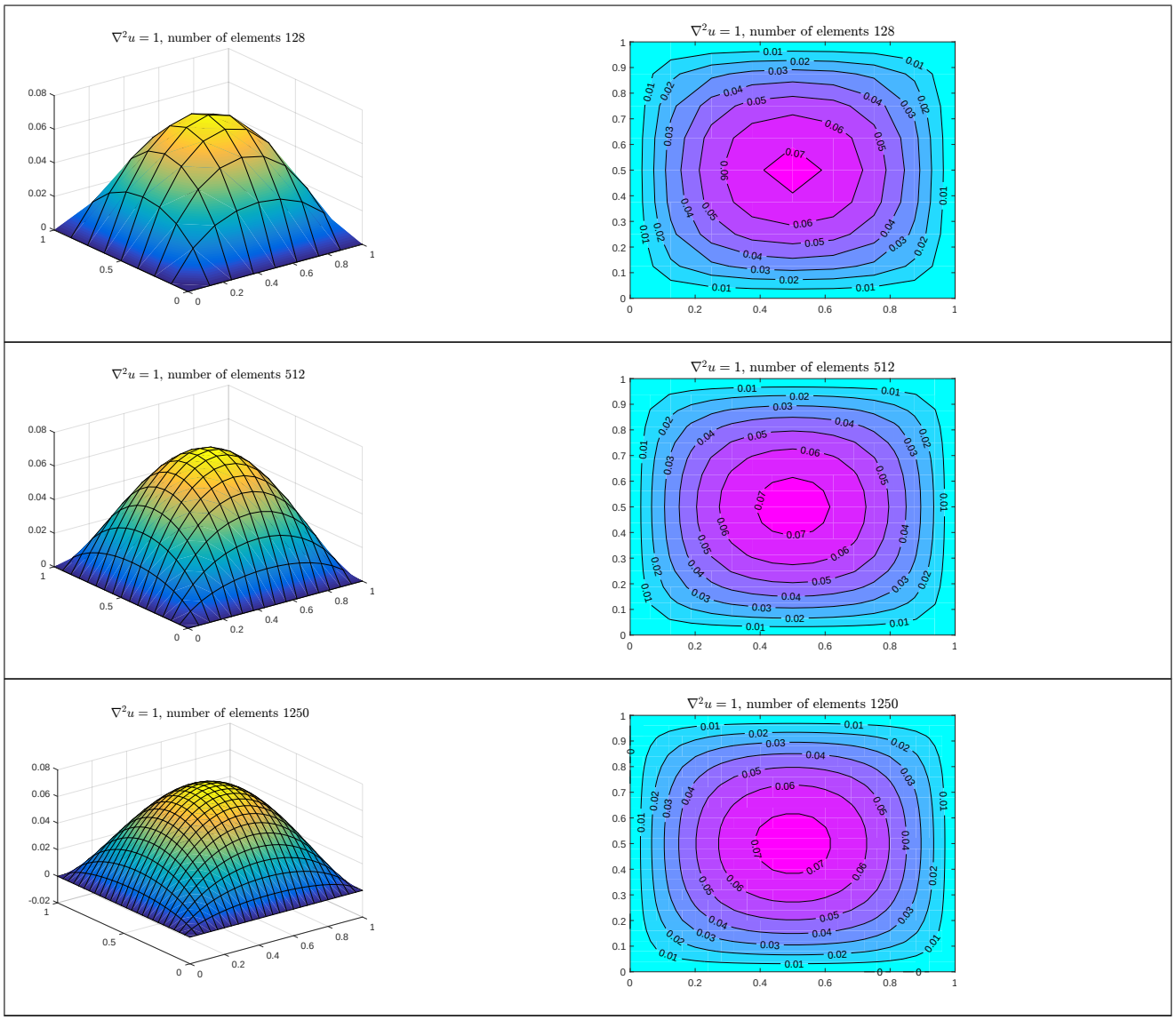
```
function A=generate_physical_coordinates_table(M,h)
%M=number_of_elements
A=zeros(M,6);
n=2*sqrt(M/2); %number of elements per row

for i=1:n/2
    for j=1:n
        ele=j+n*(i-1);
        if j==1
            A(ele,:)=[0 (i-1)*h  h (i-1)*h  0 i*h];
        else
            if mod(j,2)
                A(ele,:) =[A(ele-1,5) A(ele-1,6) ...
                 A(ele-1,5)+h A(ele-1,6)  A(ele-1,1) A(ele-1,2)];
            else
                A(ele,:) =[A(ele-1,5)+h A(ele-1,6) ...
                    A(ele-1,5) A(ele-1,6)  A(ele-1,3) A(ele-1,4)];
            end
        end
    end
end
end
```

Here is the output for the three force functions, for different number of elements.

4.16.3 case 2

Case $f(x, y) = 1$





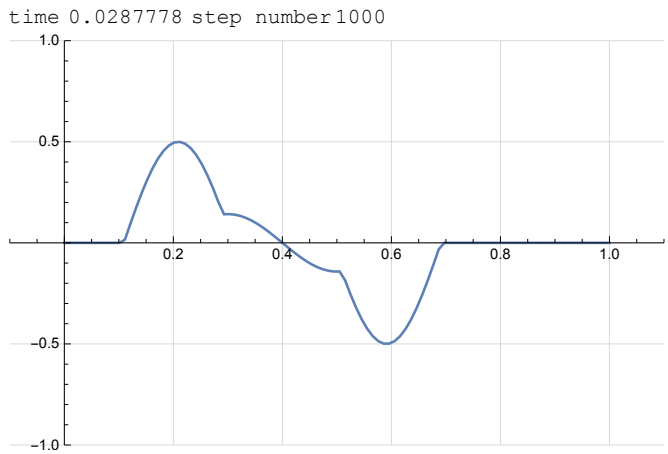
4.17 How to solve wave equation using leapfrog method?

Solve $u_{tt} = c^2 u_{xx}$ for $0 \geq x \leq 1$ with initial data defined as $f(x) = \sin\left(2\pi \frac{x-0.2}{0.4}\right)$ over $0.2 \geq x \leq 0.6$ and fixed at both ends.

Mathematica

```
SetDirectory[NotebookDirectory[]];
Dynamic[p]
```

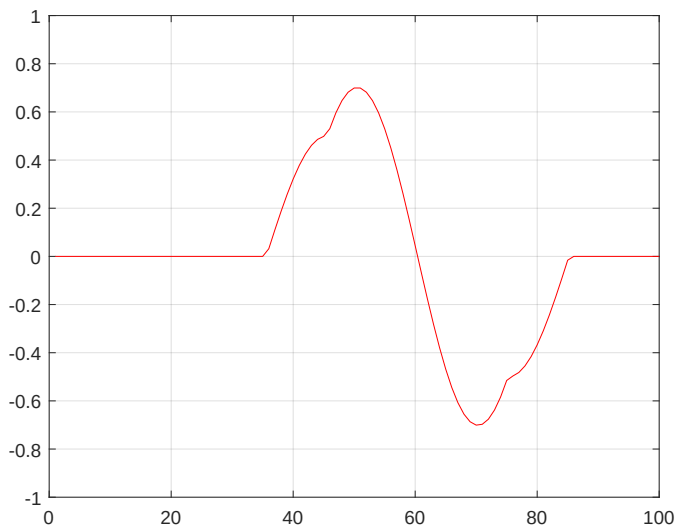
```
nPoints=100;
speed=351;
f1[x_]:= Piecewise[{{Sin[2 Pi (x-0.2)/0.4],0.2<=x<=0.6},{0,True}}];
coord = Table[j h,{j,0,nPoints-1}];
ic    = f1[#]&/@ coord;
h     = 1/(nPoints-1);
delT  = h/speed;
nSteps=1000;
end   = -1;
uNow  = ic;
uLast = uNow;
uNext = uNow;

Do[
  Which[
    n==1,
       uNext[[2;;end-1]]=(1/2)(uNow[[1;;end-2]]+uNow[[3;;end]]);
       uLast=uNow;
       uNow=uNext,

     n>2,
        uNext[[2;;(end-1)]]=uNow[[1;;(end-2)]]+uNow[[3;;end]]-uLast[[2;;(end-1)]];
        uLast=uNow;
        uNow=uNext
    ];
    (*Pause[.02];*)
    p=Grid[{
           {Row[{"time",Spacer[5],N[n*delT],Spacer[5],"step number",Spacer[2],n}]},
           {ListLinePlot[Transpose[{coord,uNow}],PlotRange->{{-.1,1.1},{-1,1}},
           ImageSize->400,GridLines->Automatic,GridLinesStyle->LightGray],SpanFromLeft}
      },Alignment->Center
  ],
   {n,1,nSteps}];

Export["images/mma_100.pdf",p]
```



Matlab

```
%Nasser M. Abbasi July 8, 2014
%solve u_tt = wave_speed * u_xx using leap frog
%  0<x<1
%  initial data is f(x)
%  initial speed = 0;
% fixed at x=0 and x=1

close all;
nPoints = 100;
speed   = 351;
f       = @(x) sin(2*pi*(x-0.2)/0.4).*(x>0.2&x<=0.6)
coord   = linspace(0,1,nPoints);
ic      = f(coord);
h       = 1/(nPoints-1);
delT    = h/speed;
last    = ic;
now     = last;
next    = now;
nSteps  = 500;

for n=1:nSteps
    if n==1
       next(2:end-1) = 1/2*(now(1:end-2)+now(3:end));
       last = now;
       now  = next;
    else
       next(2:end-1) = now(1:end-2)+now(3:end)-last(2:end-1);
```

```
        last = now;
        now  = next;
    end
    plot(now,'r');
    xlim([0 nPoints]);
    ylim([-1 1]);
    grid;
    pause(.1);
    drawnow;
end

print(gcf, '-dpdf', '-r600', 'images/matlab_100');
```



Ada

```
with Ada.Text_IO;                       use Ada.Text_IO;
with Ada.Float_Text_IO;                 use Ada.Float_Text_IO;
with Ada.Integer_Text_IO;               use Ada.Integer_Text_IO;
with Ada.Numerics;                      use Ada.Numerics;
with Ada.Numerics.Elementary_Functions; use Ada.Numerics.Elementary_Functions;

procedure foo is
   nPoints : constant Integer := 100;
   type grid_type is array (1 .. nPoints) of Float;
   Output : File_Type;
   nSteps : constant Integer := 500;
```

```
   speed  : constant Float   := 351.0;
   coord  : array (1 .. nPoints) of Float;
   h      : constant Float   := 1.0 / (Float (nPoints) - 1.0);
   delT   : constant Float   := h / speed;
   last   : grid_type        := (others => 0.0);
   now    : grid_type        := last;
   next   : grid_type        := last;

   -- output one snapshot to file to plot later
   procedure put_rec (rec : in grid_type) is
   begin
      for i in rec'Range loop
         Put (Output, rec (i));
         Put (Output,' ');
      end loop;
   end put_rec;

   -- define function of initial data
   function f (x : Float) return Float is
   begin
      if (x > 0.2 and x <= 0.6) then
         return Sin (2.0 * Pi * (x - 0.2) / 0.4);
      else
         return 0.0;
      end if;
   end f;

begin
   Create (File => Output, Mode => Out_File, Name => "output2.txt");

   for i in 1 .. nPoints loop
      coord (i) := Float (i - 1) * h;
      last (i)  := f (coord (i));
   end loop;

   now  := last;
   next := now;
   put_rec (now);
   New_Line (File => Output);

   for i in 1 .. nSteps loop
```

```
      if i = 1 then
         for j in 2 .. nPoints - 1 loop
            next (j) := (1.0 / 2.0) * (now (j - 1) + now (j + 1));
         end loop;
         last := now;
         now  := next;
         put_rec (now);
         New_Line (File => Output);
      else
         for j in 2 .. nPoints - 1 loop
            next (j) := now (j - 1) + now (j + 1) - last (j);
         end loop;
         last := now;
         now  := next;
         put_rec (now);
         if i < nSteps then
            New_Line (File => Output);
         end if;
      end if;

   end loop;

   Close (Output);
end foo;
```

The GPR file to build the above is

```
project One is

   for Main use ("foo.adb");
   for Source_Files use ("foo.adb");

   package Builder is
      for Executable_Suffix use ".exe";
   end Builder;

   package Compiler is
      for Default_Switches ("ada") use ("-g", "-gnata", "-gnato", "-fstack-check",
                                        "-gnatE", "-fcallgraph-info=su,da");
   end Compiler;
```

```
end One;
```

The animation for Ada was done by reading the output and using the plot command in Matlab to show the result.

```
close all;
A= dlmread('output2.txt');
[nRow,nCol]=size(A);
for i=1:2:nRow
    plot(A(i,:))
    ylim([-1 1]);
    grid;
    pause(.1);
    drawnow;
end
```

4.18 Numerically integrate f(x) on the real line

Problem: Integrate

$$\int_{-2}^{2} \frac{1}{5}\left(\frac{1}{100}(322+3x(98+x(37+x)))-24\frac{x}{1+x^2}\right)dx$$

The exact answer is $94/25 = 3.76$

Mathematica

```
f[x_] := (1/5)(1/100(322+3*x(98+x(37+x)))-
           24(x/(1+x^2)))
r = Integrate[f[x],{x,-2,2}]
```

```
Out[15]= 94/25
```

```
N[r]
```

```
Out[17]= 3.76
```

To compare with Matlab, replace 1 by 1.0 in the expression (or use N)

```
f[x_]:=(1.0/5)(1/100(322+3*x(98+x(37+x)))-
         24(x/(1+x^2)))
r = Integrate[f[x],{x,-2,2}];
InputForm[r]
```

```
Out[62]=  3.7600000000000007
```

Matlab

```
clear all;
format long
f=@(x)(1/5)*(1/100*(322+3*x.*(98+x.*(37+x)))...
         -24*(x/(1+x.^2)));
integral(f,-2,2)
```

```
ans =
   3.760000000000001
```

```
integral(f,-2,2,'AbsTol',1e-6,'RelTol',1e-6)
```

```
ans =
   3.760000000000001
```

4.19 Numerically integrate f(x,y) in 2D

Problem: Integrate

$$\int_{7}^{10}\int_{11}^{14}\left(x^2+4y\right)dxdy$$

The exact answer is 1719

Mathematica

```
f[x_,y_] := x^2+4 y
r = Integrate[f[x,y],{x,11,14},{y,7,10}]
```

```
Out[69]= 1719
```

Matlab

```
clear all;
format long
f=@(x,y) x.^2+4*y;
integral2(f,11,14,7,10)
```

```
ans =
     1.719000000000004e+03
```

4.20 How to solve set of differential equations in vector form

Given differential equations, written in vector form, as in

$$\begin{pmatrix} y'(t) \\ x'(t) \end{pmatrix} = \begin{pmatrix} f(t,y(t),x(t)) \\ g(t,y(t),x(t)) \end{pmatrix}$$

How to solve them?

Reference: `https://mathematica.stackexchange.com/questions/76713/dsolve-cannot-be-used-as-a-functtion`

Maple

```
restart;
A:=<1,0,0>;
```

```
p:=t-> <p1(t),p2(t),p3(t)>:
q:=t-> <q1(t),q2(t),q3(t)>:
eq1:= diff~(p(t),t) = LinearAlgebra[CrossProduct](A,q(t));
```

$$\begin{bmatrix} \frac{d}{dt} p1(t) \\ \frac{d}{dt} p2(t) \\ \frac{d}{dt} p3(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -q3(t) \\ q2(t) \end{bmatrix}$$

```
eq2:= diff~(q(t),t) = LinearAlgebra[CrossProduct](A,p(t));
```

$$\begin{bmatrix} \frac{d}{dt} q1(t) \\ \frac{d}{dt} q2(t) \\ \frac{d}{dt} q3(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -p3(t) \\ p2(t) \end{bmatrix}$$

```
ic1:= p(0)=<0.5, 0.5, 0.3>;
```

$$\begin{bmatrix} p1(0) \\ p2(0) \\ p3(0) \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.3 \end{bmatrix}$$

```
ic2:= q(0)=<0.1, 0.2, 0.3>;
```

$$\begin{bmatrix} q1(0) \\ q2(0) \\ q3(0) \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \end{bmatrix}$$

```
sol:=dsolve({eq1,eq2,ic1,ic2},convert(<p(t),q(t)>,Vector));
```

Notice in the above, the dependent variable had to be converted to one vector, hence the use of the `convert()` command. The result is

$$\begin{bmatrix} p1(t) \\ p2(t) \\ p3(t) \\ q1(t) \\ q2(t) \\ q3(t) \end{bmatrix} = \begin{bmatrix} 1/2 \\ -3/10 \sin(t) + 1/2 \cos(t) \\ 1/5 \sin(t) + 3/10 \cos(t) \\ 1/10 \\ 1/5 \cos(t) - 3/10 \sin(t) \\ 3/10 \cos(t) + 1/2 \sin(t) \end{bmatrix}$$

Mathematica

```
Clear[P, A, t]
A = {1, 0, 0}
P[t_] = {P1[t], P2[t], P3[t]}
Q[t_] = {Q1[t], Q2[t], Q3[t]}
sol = DSolve[{P'[t] == Cross[A, Q[t]], Q'[t] == Cross[A, P[t]],
         P[0] == {0.5, 0.5, 0.3}, Q[0] == {0.1, 0.2, 0.3}},
  Flatten@{P[t], Q[t]}, t]
MatrixForm@{sol}
```

$$\left(\begin{pmatrix} \mathrm{P1}(t) \to 0.5 \\ \mathrm{P2}(t) \to 0.5\cos(t) - 0.3\sin(t) \\ \mathrm{Q3}(t) \to 0.5\sin(t) + 0.3\cos(t) \\ \mathrm{P3}(t) \to 0.2\sin(t) + 0.3\cos(t) \\ \mathrm{Q2}(t) \to 0.2\cos(t) - 0.3\sin(t) \\ \mathrm{Q1}(t) \to 0.1 \end{pmatrix} \right)$$

4.21 How to implement Runge-Kutta to solve differential equations?

4.21.1 First order ODE

Solve $\frac{dx}{dt} = x(t)\sin(t)$ with initial conditions $x(0) = \frac{1}{10}$. The exact solution is $\frac{1}{10}e^{1-\cos(t)}$

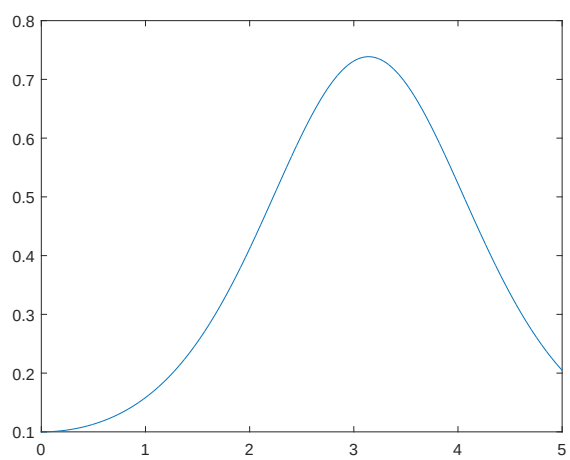
Find $x(t)$ for 5 seconds, using $\Delta_t = 0.001$

Matlab

```
function [x,t] = nma_RK4_2(  )
%solve dx/dt= x*sin(t); with x(0)=1/10 which has solution
%x(t)=1/10*exp(1-cos(t));

delT = 0.001; %time grid
t    = linspace(0,5,round(5/delT)); %time
N    = length(t);
x    = zeros(N,1);
x(1) = 0.1;  %initial conditions
for i = 1:(N-1)
    k1     = delT * f( t(i)           , x(i) );
    k2     = delT * f( t(i)+1/2*delT , x(i)+1/2*k1 );
    k3     = delT * f( t(i)+1/2*delT , x(i)+1/2*k2 );
    k4     = delT * f( t(i)+delT      , x(i)+k3 );
    x(i+1) = x(i)+1/6*(k1+2*k2+2*k3+k4);
end

  function r=f(t,x) %change RHS as needed
     r=x*sin(t);
  end
end
```

Now call the above function and plot the solution

```
[x,t]=nma_RK4_2();
plot(t,x)
```



4.21.2 second order ODE

To solve second (and higher order ODE's), we have to first rewrite the ODE as set of first order ODE and only then implement RK similarly to the above. There will be a k_i parameters for each first order ODE in the set. For this example, since we have two first order ode's (since it is second order ODE), we can call the parameters for the first oder ODE k_i and for the second order ODE z_i. For higher order, we can use a matrix to keep track of the RK parameters.

Solve $x''(t) = x(t) + x'(t)^2 - 5tx'(t)$ with initial conditions $x(0) = 2, x'(0) = 0.01$. The right hand side is $f(x, x'(t), t) = x(t) + x'(t)^2 - 5tx'(t)$. This is non-linear second order ode.

The first step is to convert this to two first order odes. Let $x = x(t)$ and $v(t) = x'(t)$, hence $x'(t) = v(t)$ and $v'(t) = x''(t) = f(x, v, t)$.

Matlab

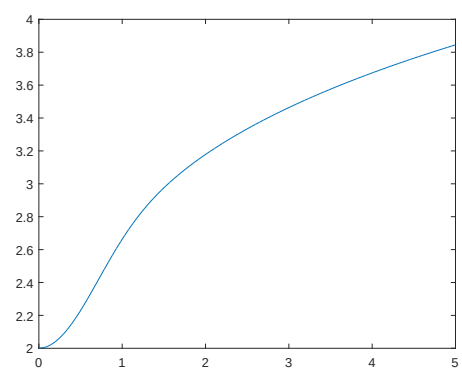
```
function [x,v,t] = nma_RK4_3(  )
%solve x''(t)=x - 5 t x'(t) + (x'(t))^2 with x(0)=2,x'(0)=0.01,

delT = 0.001; %time grid
t    = linspace(0,5,round(5/delT)); %time
N    = length(t);
x    = zeros(N,1);
v    = zeros(N,1);
x(1) = 2;  %initial conditions
v(1) = 0.01;

for i = 1:(N-1)
    k1     = delT * v(i);
    z1     = delT * f( t(i)          , x(i)        , v(i) );

    k2     = delT * (v(i) + 1/2* z1);
    z2     = delT * f( t(i)+1/2*delT , x(i)+1/2*k1 , v(i)+1/2*z1);

    k3     = delT * (v(i) + 1/2* z2);
    z3     = delT * f( t(i)+1/2*delT , x(i)+1/2*k2 , v(i)+1/2*z2);

    k4     = delT * (v(i) + z3);
    z4     = delT * f( t(i)+delT     , x(i)+k3      , v(i)+z3);

    x(i+1) = x(i)+1/6*(k1+2*k2+2*k3+k4);
```

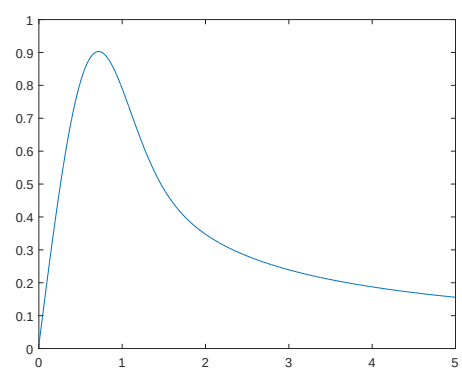
```
    v(i+1) = v(i)+1/6*(z1+2*z2+2*z3+z4);
end

  function r=f(t,x,v)
     r=x - 5*t*v + v^2;
  end
end
```

The above function is called the solution is plotted

```
[x,v,t]=nma_RK4_3();
plot(t,x);
plot(t,v);
```

Here is plot result





A better way to do the above, which also generalized to any number of system of equations is to use vectored approach. Writing

$$[\dot{x}] = [f(t, x, x', \dots)]$$

Where now $[x]$ is the derivative of state vector and $[f(t, x, x', \dots)]$ is the right hand

side, in matrix form. The above second order is now solved again using this method. This shows it is simpler approach than the above method.

```
function [x,t] =nma_RK4_4(  )
  %solve x''(t)=x - 5 t x'(t) + (x'(t))^2 with x(0)=2,x'(0)=0.01,

  delT = 0.001; %time grid
  t    = linspace(0,5,round(5/delT)); %time
  N    = length(t);
  x     = zeros(2,N);
  x(1,1) = 2;  %initial conditions
  x(2,1) = 0.01;

  for i = 1:(N-1)
      k1     = delT * f( t(i)           , x(:,i));
      k2     = delT * f( t(i)+1/2*delT , x(:,i)+1/2*k1);
      k3     = delT * f( t(i)+1/2*delT , x(:,i)+1/2*k2);
      k4     = delT * f( t(i)+delT      , x(:,i)+k3);

      x(:,i+1) = x(:,i)+1/6*(k1+2*k2+2*k3+k4);
  end

    function r=f(t,x)
       r=[x(2)
          x(1)-5*t*x(2)+x(2)^2];
    end
  end
```

The above is called as follows

```
[x,t]=nma_RK4_4();
plot(t,x(1,:)) %plot the position x(t) solution
plot(t,x(2,:)) %plot the velocity x'(t) solution
```

Same plots result as given above.

4.22 How to differentiate treating a combined expression as single variable?

Problem: Differentiate

$$e^{\frac{2r}{\sqrt{a}}}+3\left(\frac{r}{\sqrt{a}}\right)+\left(\frac{r}{\sqrt{a}}\right)^2$$

w.r.t $\frac{r}{\sqrt{a}}$ to produce

$$2e^{\frac{2r}{\sqrt{a}}}+3+2\frac{r}{\sqrt{a}}$$

In other words, we want to treat $\frac{r}{\sqrt{a}}$ as x in the expression

$$e^{2x}+3x+x^2$$

Mathematica

```
Clear[p, x, r, a]
p[x_] := Exp[2 x] + x^2 + 3*x;
v = r/Sqrt[a];
With[{v = x}, Inactive[D][p[v], v]];
Activate[%];
% /. x -> v
```

$$\frac{2r}{\sqrt{a}}+2e^{\frac{2r}{\sqrt{a}}}+3$$

Maple

Credit for the Maple answer goes to an internet post by Carl Love

```
restart;
D(x->exp(2*x)+3*x+x^2) (r/sqrt(a));
```

$$2\,\mathrm{e}^{2\,\frac{r}{\sqrt{a}}}+3+2\,\frac{r}{\sqrt{a}}$$

4.23 How to solve Poisson PDE in 2D with Neumann boundary conditions using Finite Elements

Solve $\nabla^2 T - 20T = -20$ on the unit square $0 \leq x \leq 1, 0 \leq y \leq 1$ subject to insulated boundary conditions at $x = 0$ and $y = 0$ and $T = 0$ at $x = 1, y = 1$.

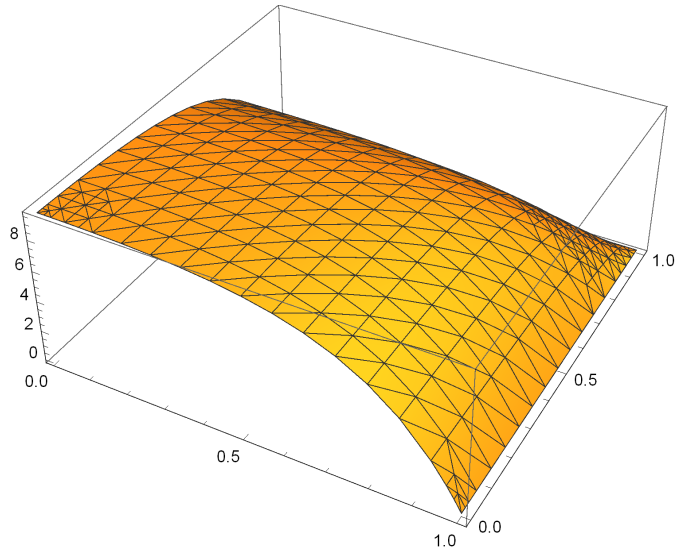
4.23.1 Mathematica

In Mathematica 10.0, one can use Finite elements directly. Here is the code to solve the above.

```
Clear[u,x,y];
r=NDSolveValue[{D[u[x,y],{x,2}]+D[u[x,y],{y,2}]-20*u[x,y]==
    -200+NeumannValue[0,x==0]
    +NeumannValue[0,y==0],
    DirichletCondition[u[x,y]==0,x==1],
    DirichletCondition[u[x,y]==0,y==1]},
    u,{x,0,1},{y,0,1},
    Method->{"FiniteElement",
      "MeshOptions"->{"BoundaryMeshGenerator"->"Continuation"}}];
```

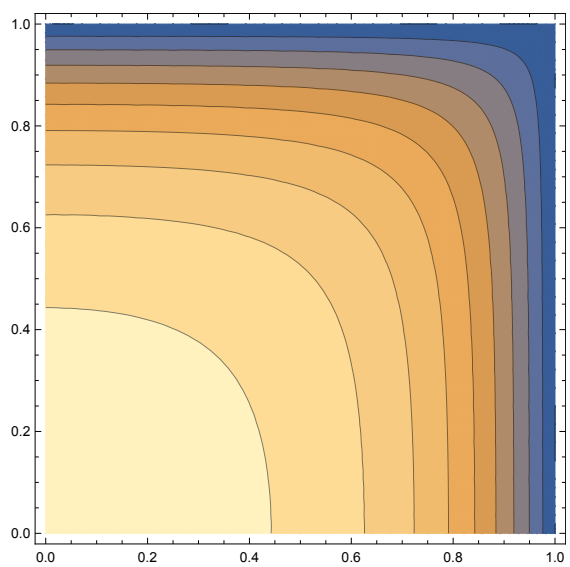
Plotting the solution

```
Plot3D[r[x,y],{x,0,1},{y,0,1},Mesh->All]
```



Now we make a contour plot

```
ContourPlot[r[x,y],{x,0,1},{y,0,1}]
```



CHAPTER 5

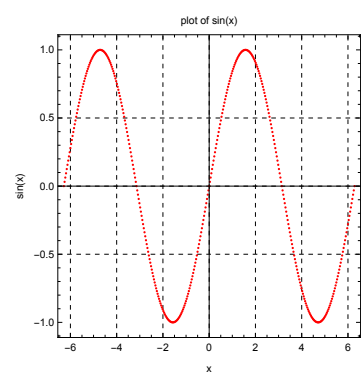
PLOTTING HOW TO, ELLIPSE, 3D

5.1 Generate a plot using x and y data

Problem: Generate x and y data for $sin(x)$ and plot the data.

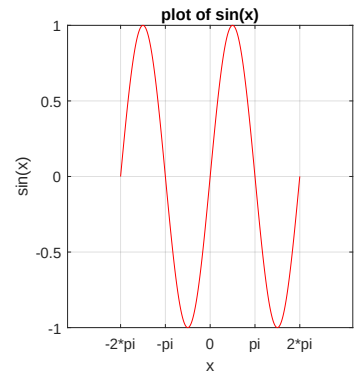
Mathematica

```
Remove["Global`*"];
data = Table[{x,Sin[x]},
        {x, -2Pi,2 Pi,Pi/100}];
ListPlot[data,
     Frame -> True,
     GridLines->Automatic,
     GridLinesStyle->Dashed,
     ImageSize->300,
     AspectRatio->1,
     PlotStyle->Red,
     FrameLabel->{{"sin(x)",None},
        {"x","plot of sin(x)"}},
     FrameTicks->{{-2Pi,-Pi,0,Pi,2Pi},
                None,Automatic,None}
     ]
```



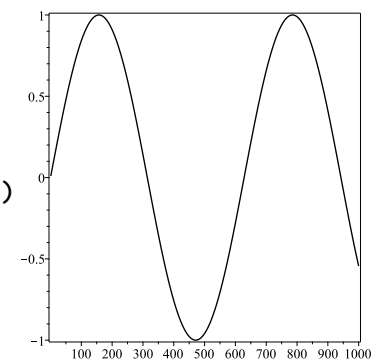
Matlab

```
clear all;
x = -2*pi:pi/100:2*pi;
plot(x,sin(x),'r');
grid on;
title('plot of sin(x)');
xlabel('x');
ylabel('sin(x)');
set(gca,'XTick',-2*pi:pi:2*pi)
set(gca,'XTickLabel',...
   {'-2*pi','-pi','0','pi','2*pi'})
set(gcf,'Position',[10,10,340,340]);
```



Maple

```
restart; with(plots);
lis := [seq(sin((1/100)*theta), theta = 1 .. 1000)
listplot(lis, axes = box)
```



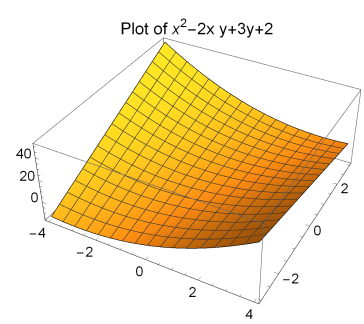
5.2 Plot the surface described by 2D function

Problem: Plot the function

$$f(x,y) = x^2 - 2xy + 3y + 2$$

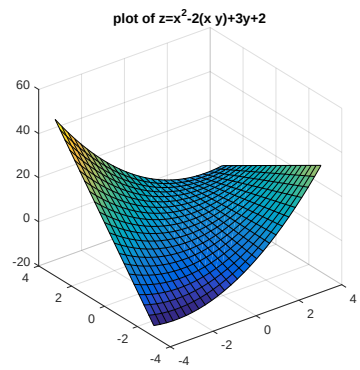
Mathematica

```
Remove["Global`*"];
f[x_,y_]:=x^2-2x y+3y+2;
Plot3D[f[x,y],{x,-4,4},{y,-3,3},
    PlotLabel->"Plot of x^2-2x y+3y+2",
    ImageSize->300]
```



Matlab

```
clear all; close all;
x = -4:.3:4;
y = -3:.3:3;
[X,Y] = meshgrid(x,y);
Z = X.^2 - 2*(X.*Y) + 3*Y + 2;
surf(X,Y,Z)
title('plot of z=x^2-2(x y)+3y+2');
set(gcf,'Position',[10,10,420,420]);
```



5.3 Find the point of intersection of 3 surfaces

Problem: Given 3 surfaces find the point where they intersect by solving the linear system and display the solution.

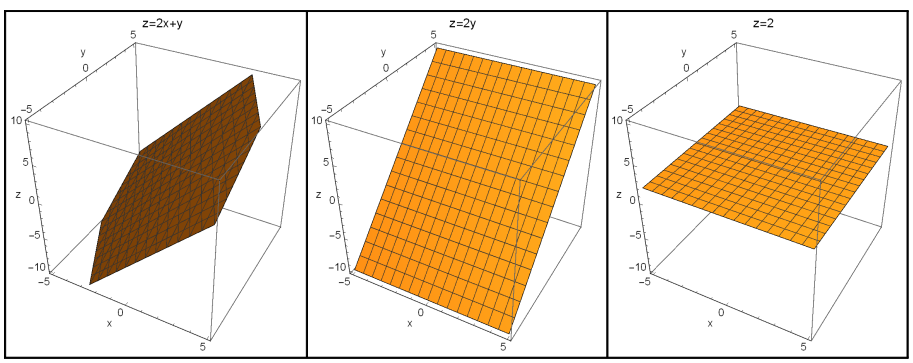
Let the 3 surfaces be

$$
\begin{aligned}
z &= 2x + y \\
z &= 2y \\
z &= 2
\end{aligned}
$$

In Mathematica, plot the surfaces using the Plot3D command, and then use LinearSolve to solve the system of equations in order to find point of intersection, then use ScatterPlot to plot the point solution. In Matlab, use surf command to do the plotting, and solve the system of equations using Matlab

Mathematica

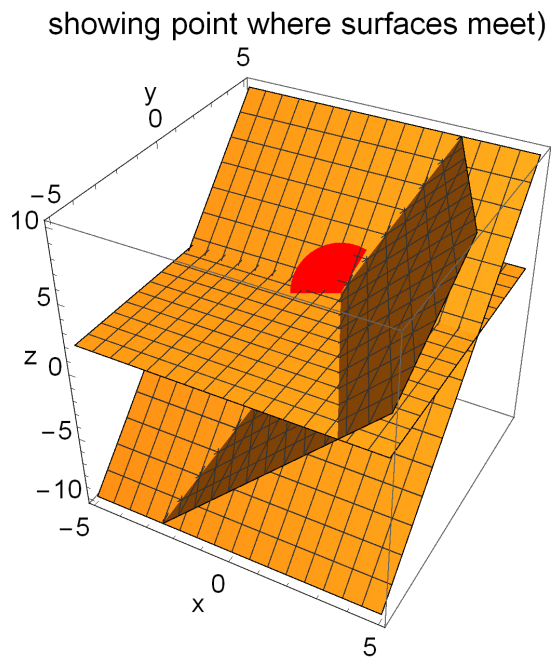
```
Remove["Global`*"];
eq1 = z-2x-y ==0;
eq2 = z-2y ==0;
eq3 = z- 2 ==0;
p1 = ContourPlot3D[Evaluate@eq1,
         {x,-5,5},{y,-5,5},{z,-10,10},
         AxesLabel->{"x","y","z"},
         PlotLabel->"z=2x+y",
         ImageSize->250];
p2 = ContourPlot3D[Evaluate@eq2,
        {x,-5,5},{y,-5,5},{z,-10,10},
        AxesLabel->{"x","y","z"},
        PlotLabel->"z=2x",
        ImageSize->250];
p3 = ContourPlot3D[Evaluate@eq3,
        {x,-5,5},{y,-5,5},{z,-10,10},
        AxesLabel->{"x","y","z"},
        PlotLabel->"z=2",
        ImageSize->250];
Grid[{{p1,p2,p3}},Frame->All]
```




```
{b,a} = CoefficientArrays[{eq1,eq2,eq3},
         {x,y,z}];
sol   = LinearSolve[a,-b]//N
```

```
Out[301]= {0.5,1.,2.}
```

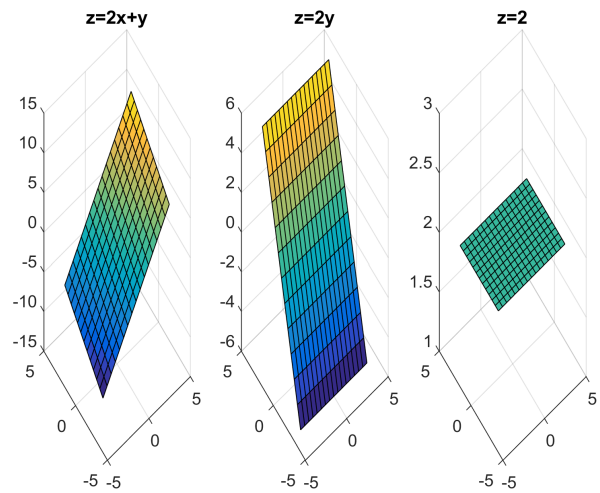
```
Show[p1,p2,p3,
  Graphics3D[{PointSize[0.2],
       Red,Point[sol]}],
  PlotLabel->"point where surfaces meet)"]
```

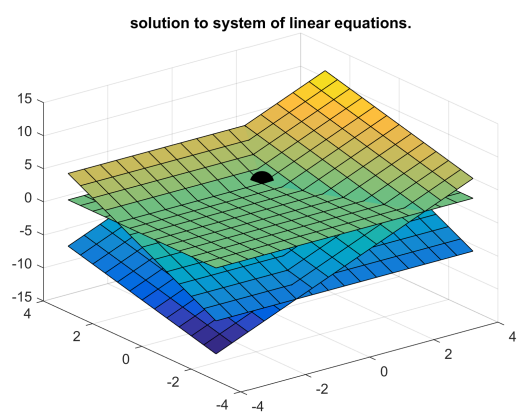


Matlab

```
clear all; close all;
x = -4:.5:4;
y = -3:.5:3;

[X,Y] = meshgrid(x,y);

subplot(1,3,1);

Z = 2*X + Y;  surf(X,Y,Z);
title('z=2x+y');

subplot(1,3,2);
Z = 2*Y;  surf(X,Y,Z);
title('z=2y');

subplot(1,3,3);
Z = zeros(length(y),length(x));
Z(1:end,1:end)=2;
surf(X,Y,Z);
title('z=2');

A=[-2 -1 1;0 -2 1;0 0 1];
b=[0;0;2];
sol=A\b

figure;
surf(X,Y,2*X + Y); hold on;
surf(X,Y,2*Y);
surf(X,Y,Z);

%now add the point of intersection
%from the solution

plot3(sol(1),sol(2),sol(3),...
      'k.','MarkerSize',30)

title('solution to system of linear equations.');
```

```
sol =

    0.5000
    1.0000
    2.0000
```





5.4 How to draw an ellipse?

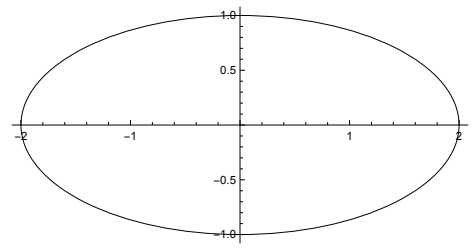
5.4.1 draw an ellipse centered at origin and given its semi-major/minor (a, b)

draw an ellipse centered at $(0, 0)$ with $a = 2, b = 1$ being its semi-major and minor.

The ellipse equation for the above is $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

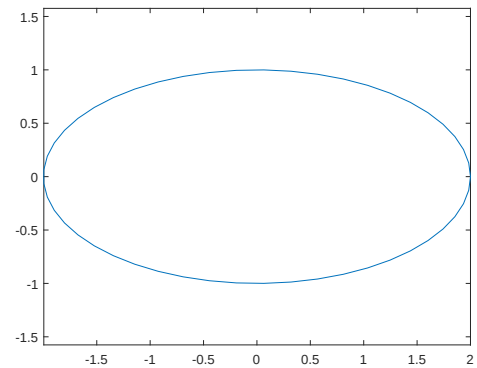
Mathematica

```
a = 2;
b = 1;
Graphics[Circle[{0, 0}, {a, b}], Axes -> T
```



Matlab

```
a=2;
b=1;
t=linspace(0,2*pi,50);
plot(a*cos(t),b*sin(t))
axis equal
```

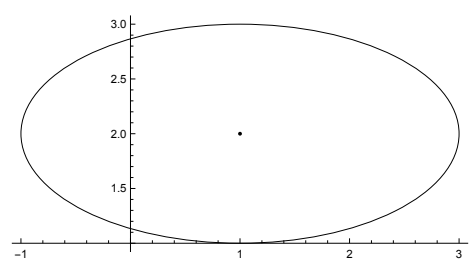


5.4.2 draw an ellipse centered at (x, y) and given its semi-major/minor (a, b)

draw an ellipse centered at $(1, 2)$ with $a = 2, b = 1$ being its semi-major and minor. The ellipse equation for the above is $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

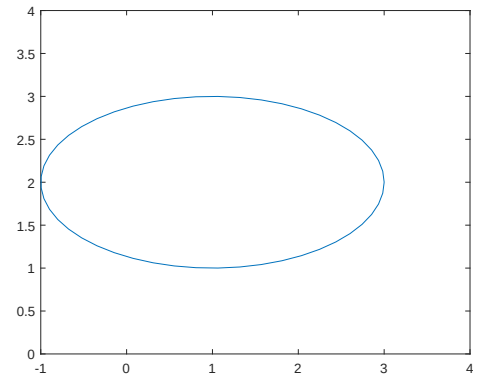
Mathematica

```
a = 2;
b = 1;
Graphics[
 {
  Circle[{1, 2}, {a, b}],
  Point[{1, 2}]
  }, Axes -> True
 ]
```



Matlab

```
close all;
a=2; b=1;
t=linspace(0,2*pi,50);
plot(a*cos(t)+1,b*sin(t)+2)
axis equal
xlim([-1,4]);
ylim([0,4]);
```



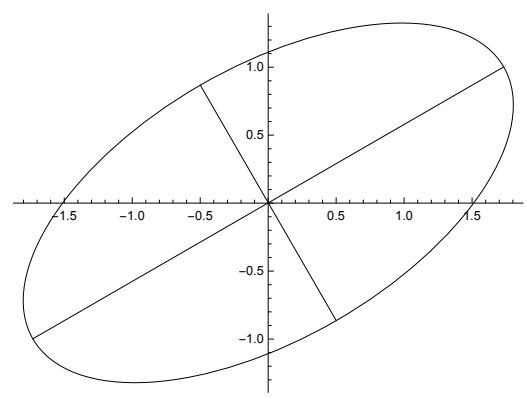
5.4.3 draw an ellipse centered at origin and given its semi-major/minor (a, b) and rotated at angle θ

draw an ellipse centered at $(0, 0)$ with $a = 2, b = 1$ being its semi-major and minor titlted at angle $\theta = 30$ degrees anti-clockwise.

The ellipse equation for the above is $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. After drawing it, we rotate it.

Mathematica

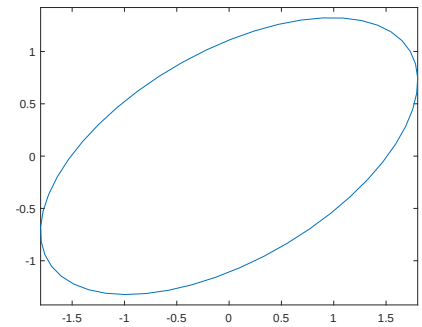
```
a = 2;
b = 1;
Graphics[
 Rotate[
  {Circle[{0, 0}, {a, b}],
   Line[{{-a, 0}, {a, 0}}],
   Line[{{0, -b}, {0, b}}]
   }
  , 30 Degree
  ]
 , Axes -> True
 ]
```



Matlab

Matlab does not have rotate function, so use the rotation tranformation matrix.

```
close all;
a   = 2;
b   = 1;
t   = linspace(0,2*pi,50);
rotateAngle = 30*pi/180;
mat = [cos(rotateAngle) -sin(rotateAngle)
       sin(rotateAngle) cos(rotateAngle)];

%2x2 times 2x50 = 2x50
data = mat*[a*cos(t);b*sin(t)];
plot(data(1,:),data(2,:))
axis equal
```



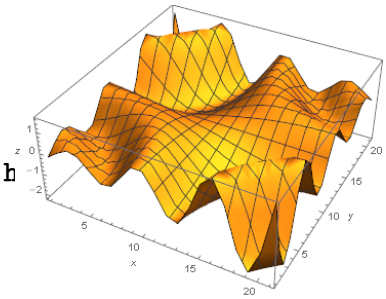
5.5 How to plot a function on 2D using coordinates of grid locations?

Sometimes in finite difference we want to evaluate the force or the function in the RHS of the equation $f(x,y)$ using the coordinates x,y on a 2D grid. The grid spacing can be h and the coordinates extend from some x value to some other x value and from some y value to some other y values. In Matlab, the coordinates are setup using `meshgrid`.

In Mathematica, there is really no need for `meshgrid` as the build-in command `Map` can be used to evaluate the function at each grid point. The coodinates physical values are generated using the `Table` command. Here is example evaluating and plotting $f(x,y) = \sin(10xy)e^{-xy}$ on a grid of spacing $h = 1/N$ where N is number of elements. The grid goes from $x = -1$ to $x = 1$ by spacing of h. Same for the y direction.

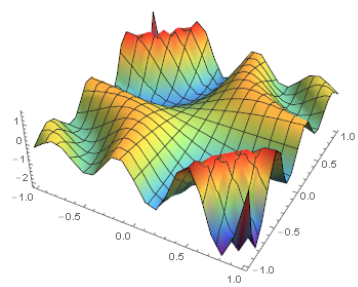
Mathematica

```
Clear[x, y, z];
nElem = 10.;
h = 1/nElem
N@Range[-1, 1, h]
grid = N@Table[{i, j}, {i, -1, 1, h}, {j, -1, 1, h
f[x_, y_] := Sin[x*10 y] Exp[-x y];
force = Map[f[#[[1]], #[[2]]] &, grid, {2}];
ListPlot3D[force, AxesLabel -> {x, y, z}]
```



To use the actual physical coordinates on the axis above, then replace the plot above with this below. This also includes an implementation of meshgrid like function in Mathematica which makes it easier to work with for someone familiar with Matlab:

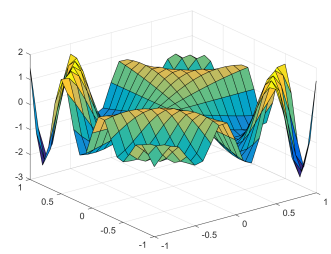
```
meshgrid[x_List,y_List]:={ConstantArray[x, Length[x]],
  Transpose@ConstantArray[y, Length[y]]};

nElem = 10.;
h = 1/nElem;
{x, y} = meshgrid[Range[-1, 1, h], Range[-1, 1, h]
f[x_, y_] := Sin[x*10 y] Exp[-x y];
force = f[x, y];
pts = Flatten[{x, y, force}, {2, 3}];
ListPlot3D[pts, PlotRange -> All,
      AxesLabel -> Automatic, ImagePadding -> 20,
      ColorFunction -> "Rainbow", Boxed -> False]
```



To see each point value, use `InterpolationOrder -> 0` in the plot command.

Matlab

```
clear all; close all;
nElem = 10;
h = 1/nElem;
[X,Y] = meshgrid(-1:h:1,-1:h:1);
f = @(x,y) sin(x*10.*y) .* exp(-x.*y);
force = f(X,Y);
surf(X,Y,force)
```

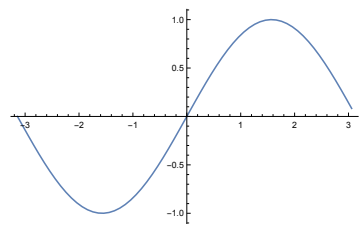


5.6 How to make table of x, y values and plot them

Generate a list or table of $(x, \sin(x))$ values and plot them.

Mathematica

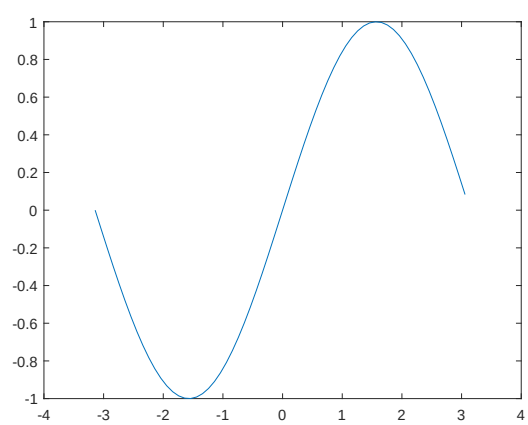
```
lst=Table[{x, Sin[x]}, {x, -Pi, Pi, 1/10}];
ListLinePlot[lst]
```



Matlab

Matlab does not have a nice function like `Table` so one can either make the x variable and then use it to make $\sin(x)$ or use `arrayfun`

```
close all;
x=-pi:.1:pi;
plot(x,sin(x));
```



Using `arrayfun`

```
A=cell2mat(arrayfun(@(x) [x;sin(x)],..
       -pi:.1:pi, 'UniformOutput',false
plot(A(1,:),A(2,:))
```

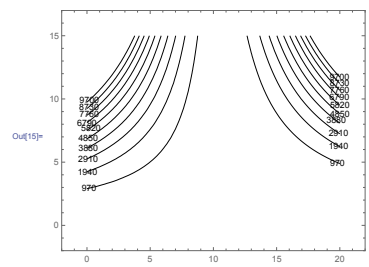
5.7 How to make contour plot of $f(x, y)$?

Problem: Make contour of $f(x, y) = (11 - x - y)^2 + (1 + x + 10y - xy)^2$ over region $0 < x < 20$ and $0 < y < 10$.

Contour label placement seems to be better in Matlab. It is put in the middle of the line automatically.

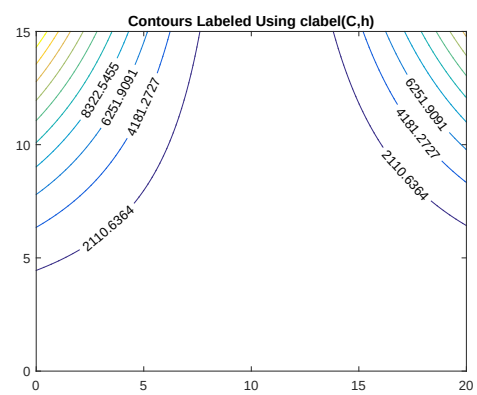
Mathematica

```
f = (11 - x - y)^2 + (1 + x + 10 y - x y)^2;
ContourPlot[f, {x, 0, 20}, {y, 0, 15},
 Contours -> 10, ContourShading -> None,
 ContourLabels -> Function[{x, y, z},
   Text[z, {x, y}]], AspectRatio -> Automatic,
   PlotRangePadding -> 2]
```



Matlab

```
x = 0:0.05:20;
y = 0:0.05:15;
[x,y] = meshgrid(x,y);
z = (11-x-y).^2 + (1+x+10*y-x.*y).^2;
[C,h] =contour(x,y,z,10);
clabel(C,h)
title('Contours Labeled Using clabel(C,h)'
```



CHAPTER 6

Some symbolic operations

6.1 How to find all exponents of list of expressions?

Given $\{f^2, g^3, k^p, h\}$ write code to return $\{2, 3, p, 1\}$ as the exponents.

Mathematica

```
ClearAll[f, g, k, p, h];
lst = {f^2, g^3, k^p, h};
Cases[lst, x_^n_. :> n]
```

```
{2, 3, p, 1}
```

Maple

```
unassign('f,g,k,p,h');
foo:=proc(expr)
     if nops(expr)=1 then
        return 1;
     elif hastype(expr,`^`) then
        return( op(2,expr) );
     fi;
     return(NULL);
end proc:

expr:=[f^2, g^3, k^p, h]:
map(foo,expr)
```

```
[2, 3, p, 1]
```

This document was compiled using

Text Font: ec-mlmr12

Math Operator Font: rm-mlmr12

Math Letter Font: mlmmi12